Universidad
Carlos III de Madrid

Departamento de Telemática

PROYECTO FIN DE CARRERA

Marco Genérico para el Desarrollo de Aplicaciones Migrables entre Windows Mobile y Android

Autor: Ricardo Fraile Martínez

Tutor: Pablo Basanta Val

Leganés, 18 de Julio de 2013

Título: Marco Genérico para el Desarrollo de Aplicaciones Migrables entre Windows Mobile y Android.
Autor: Ricardo Fraile Martínez
Director: Pablo Basanta Val

EL TRIBUNAL

Presidente: Mario Muñoz Organero

Vocal: Francisco Valera Pintor

Secretario: Matilde Sánchez Fernández

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 18 de Julio de 2013 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar, quisiera agradecer a mi tutor Pablo, sus consejos y orientación a lo largo de todo el proyecto, han sido clave para que éste llegase a su fin. Además quisiera agradecer la infinita paciencia que ha tenido conmigo.

También quisiera agradecer a mi familia y en especial a mi mujer, haberme apoyado y animado tanto durante todo este tiempo, no saben lo mucho que me han ayudado.

Finalmente, quisiera agradecer a los amigos que me han acompañado durante estos años de universidad porque no hubiera sido lo mismo sin ellos.

Resumen

Este proyecto surge de la necesidad de las empresas de abaratar los costes de desarrollo de las aplicaciones móviles en distintos sistemas operativos, concretamente Windows Mobile y Android.

Para ello se diseña un marco genérico para el desarrollo de aplicaciones que permita desarrollar aplicaciones en una plataforma (Windows Mobile) y migrarla fácilmente a otra (Android) aplicando unas reglas de transformación. Para comprobar la utilidad del marco se ha desarrollado una aplicación ejemplo utilizando el marco genérico basada en el Mundial de futbol del 2010 y se ha procedido a migrarla. Finalmente se han realizado pruebas sobre la aplicación para comprobar la eficacia del marco diseñado.

Palabras clave: SmartPhone, Windows Mobile, Android

Abstract

This project arises from the need for companies to reduce costs of developing mobile applications on different operating systems, specifically Windows Mobile and Android.

To do this, a generic application development framework has been designed for the development of applications on one platform (Windows Mobile) and easily migrate to another one(Android) by applying transformation rules. To check the usefulness of the framework a sample application based on the soccer World Cup 2010 has been developed using the generic framework and has been proceeded to migrate it. Finally, the application has been tested to verify the effectiveness of the designed framework.

Keywords: SmartPhone, Windows Mobile, Android

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción y objetivos

1.1 Introducción

En la actualidad tenemos numerosos sistemas operativos móviles, cada uno con lenguajes y herramientas de desarrollo propias. Para las empresas el coste de desarrollar aplicaciones para todas las plataformas supone muchas veces un coste elevado, equipos independientes que desarrollan las aplicaciones, por ello este proyecto pretende diseñar e implementar un marco genérico para el desarrollo de aplicaciones móviles que se pueda aplicar a todas las plataformas.

Teniendo en cuenta esta idea, el proyecto se va a centrar en las dos plataformas que dominaban el mercado de los smartphones en el 2010, Windows Mobile y Android. A finales del 2010 la subida en la venta de smartphones fue de un 96%, con una cuota de mercado del 19,6%, que aumenta exponencialmente año tras año ([1]).

La mayoría de las aplicaciones del mercado (deportivas, noticias, transportes, moda, realidad aumentada) se basan en la premisa de recuperación y presentación de datos al usuario, por este motivo, este proyecto se va a centrar en diseñar e implementar un modelo de estas características.

Teniendo en cuenta que el año 2010 tiene un acontecimiento deportivo importante, el mundial de fútbol ([2]), se va a desarrollar una aplicación con este motivo para realizar las pruebas del marco genérico que se pretende diseñar.

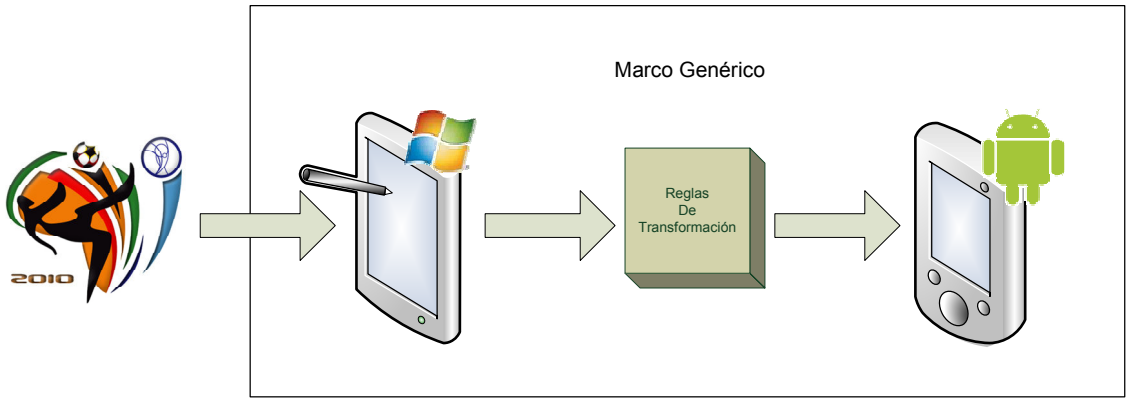

Figura 1: Esquema del proyecto

1.2 Objetivos del Proyecto

El objetivo del proyecto es el diseño e implementación de un marco genérico para el desarrollo de aplicaciones móviles, aplicárselo al desarrollo de una aplicación en la plataforma Windows Mobile y migrarlo usando unas reglas de transformación a la plataforma Android.

De esto se deducen los siguientes objetivos parciales:

1- **Diseño del marco genérico**

Esto es, diseñar una aplicación para la recuperación y presentación de datos en aplicaciones móviles. Se diseñara una serie de funciones que deberá soportar dicho marco.

2- **Implementación del marco genérico**

Implementación de las funciones que debe cumplir el marco en Android y Windows Mobile, y las reglas de transformación que vamos a necesitar para poder migrar el código entre plataformas.

3- **Implementación de la aplicación**

Se implementará una aplicación ejemplo en Windows Mobile y se procederá a migrar dicha aplicación a Android, utilizando para ello Eclipse con la SDK de Android y Visual Studio 2008 profesional con la SDK de Windows Mobile 6.5. Esta aplicación se basará en el Mundial de fútbol 2010.

4- **Rendimiento**

Se utilizarán los emuladores y dispositivos de las plataformas Android y Windows Mobile 6.5 para comprobar el funcionamiento y rendimiento de la aplicación en ambas plataformas.

1.3 Estructura de la Memoria

Para el desarrollo de los objetivos parciales la memoria tiene esta estructura:

- Bloque I: Introducción
    - Capítulo 1. Introducción y objetivos.
- Bloque II: Estado del arte

  Análisis de las diferentes tecnologías involucradas en diseño e implementación de aplicaciones para Android y Windows Mobile
    - Capítulo 2. Herramientas para el desarrollo para Windows Mobile
    - Capítulo 3. Herramientas para el desarrollo para Android
- Bloque III: Diseño e implementación del marco genérico

  Diseño del marco genérico, de la aplicación, implementación del diseño y pruebas para comprobar la viabilidad del proyecto.
    - Capítulo 4. Diseño e Implementación del marco genérico.
        - Requisitos
        - Diagrama de alto nivel
        - Diagramas de implementación
            - Android
            - Windows Mobile
    - Capítulo 5. Caso de Uso: Aplicación del mundial
        - Implementación de la aplicación en las diferentes tecnologías.
        - Pruebas y rendimiento
- Bloque IV: Conclusiones y futuras líneas de trabajo

  Resultado y conclusiones sobre la consecución del proyecto para indicar el grado de satisfacción de los objetivos.
    - Capítulo 6. Conclusiones y líneas futuras de trabajo.
- Bloque V: Glosario
- Bloque VI: Referencias e hiperenlaces

2. Herramientas: Desarrollo en Windows Mobile

2.1 Introducción

Windows Mobile ([3]) es un sistema operativo móvil compacto desarrollado por Microsoft, y diseñado para su uso en teléfonos inteligentes (Smartphones) y otros dispositivos móviles. Se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows.

Figura 2: Logo Windows Mobile

Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente. Además, existe una gran oferta de software de terceros disponible para Windows Mobile, la cual se puede adquirir a través de Windows Marketplace for Mobile o simplemente descargando los archivos autoinstalables "`.cab`" en el dispositivo.

Originalmente apareció bajo el nombre de Pocket PC, como una ramificación de desarrollo de Windows CE para equipos móviles con capacidades limitadas. En la actualidad, la mayoría de los teléfonos con Windows Mobile vienen con un estilete digital, que se utiliza para introducir comandos pulsando en la pantalla. Windows Mobile ha evolucionado y cambiado de nombre varias veces durante su desarrollo, siendo la última versión la llamada Windows Mobile 6.5. A diferencia de lo que muchos piensan, Windows Phone 7 no pertenece a esta familia, puesto que es un sistema operativo creado de 0 sin base ni compatibilidad con Windows CE.

La versión 6.5 es una actualización importante de la plataforma Windows Mobile que fue liberada a los fabricantes el 11 de mayo de 2009. El 6 de octubre de 2009 fue el lanzamiento mundial de esta nueva versión de Windows Mobile que a partir de ese día se conoce también por Windows Phone. La mayor novedad de Windows Mobile 6.5 fue el cambio completo de la interfaz de usuario para adaptarlo a los nuevos dispositivos táctiles de forma que se pudieran manejar fácilmente con el dedo, sin necesidad de un puntero como en versiones anteriores.

Algunas novedades importantes fueron:

- Windows Marketplace: A partir de la versión 6.5, todos los teléfonos incorporan un acceso a la tienda de aplicaciones de Microsoft.

- Internet Explorer Mobile 6: Nueva versión de Internet Explorer que había sido reescrito completamente para proporcionar una navegación más intuitiva. Se actualizó su interfaz para poder ser controlado en dispositivos táctiles de forma fluida.
- Microsoft My Phone: Esta aplicación permitía disponer de 200 MB en los servidores de Microsoft para mantener una copia de seguridad de los datos del teléfono móvil como contactos, mensajes, SMS, notas, documentos, y música.
- Microsoft Office Mobile 6.1: Contenía los siguientes programas: Word Mobile, Excel Mobile, PowerPoint Mobile y OneNote Mobile que son versiones de las aplicaciones Office adaptadas a un teléfono móvil. Esta versión de Office es capaz de trabajar directamente con ficheros con el formato estándar de Open XML que está implementado desde la versión Office 2007.

2.2 Lenguaje de programación – C#

C# (pronunciado si sharp en inglés) ([4]) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes. A continuación se puede observar el típico ejemplo de “Hola Mundo” por el que todos aprendemos nuevos lenguajes:

```
public class HelloWorld
{
   public static void Main()
   {
      System.Console.WriteLine("Hello, World!");
   }
}
```

El nombre C Sharp fue inspirado por la notación musical, donde ♯ (sostenido, en inglés sharp) indica que la nota (C es la nota do en inglés) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' viene de cuatro '+' pegados.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono, el cual genera programas para distintas plataformas como Windows, Unix y GNU/Linux.

2.3 Net Compact Framework 2.0

Microsoft. NET Compact Framework (.NET CF) ([5]) es una versión del .NET Framework que está diseñado para funcionar en Windows CE, es decir, en móviles/dispositivos embebidos tales como PDAs, teléfonos móviles, controladores de fábrica o set-top boxes. NET Compact Framework contiene algunas de las bibliotecas de clases que las utiliza .NET Framework, pero también añade algunas bibliotecas diseñadas específicamente para dispositivos móviles, como Windows CE InputPanel. Sin embargo, las bibliotecas no son copias exactas de .NET Framework, normalmente se escalan hacia abajo y se limitan para ocupan menos espacio.

Es posible desarrollar aplicaciones que utilizan .NET Compact Framework en Visual Studio.NET 2003, en Visual Studio 2005, en Visual Studio 2008, en C# o Visual Basic.NET. Las aplicaciones resultantes están diseñadas para ejecutarse en un compilador JIT especiales, dispositivos móviles, de alto rendimiento.

.NET Compact Framework se puede ejecutar en equipos de escritorio usando .NET Framework, pero su interfaz de usuario no se puede actualizar para parecerse a la de una aplicación desarrollada para PC de escritorio. Microsoft.

Una versión del. NET Compact Framework también está disponible para la Xbox 360. Si bien las características del tiempo de ejecución es igual a al .NET CF normal, sólo un subconjunto de la biblioteca de clases está disponible. Esta versión es utilizada por XNA Framework para ejecutar los juegos gestionados en la consola. Hay otras limitaciones, como la limitación 256. Windows Phone 7 se crea a partir de esta idea, por lo que una de las maneras de programación es a través de XNA.

2.4 Windows Mobile 6 y 6.5 Developer Toolkit

Windows Mobile 6 Developer Toolkit añade documentación, librerías, ejemplos en código, y emuladores a Visual Studio que permite la creación de aplicaciones para Windows Mobile 6.

Windows Mobile 6.5 Developer Toolkit añade documentación, librerías, ejemplos en código, y emuladores a Visual Studio que permite la creación de aplicaciones para Windows Mobile 6.5. Es una actualización de Windows Mobile 6 que añade reconocimiento de gestos y emuladores de los nuevos dispositivos que aparecían en el mercado con resoluciones de pantalla más grandes como WVGA.

En la Figura 3 podemos observar una imagen del emulador de Windows Mobile funcionando sobre Windows 7:

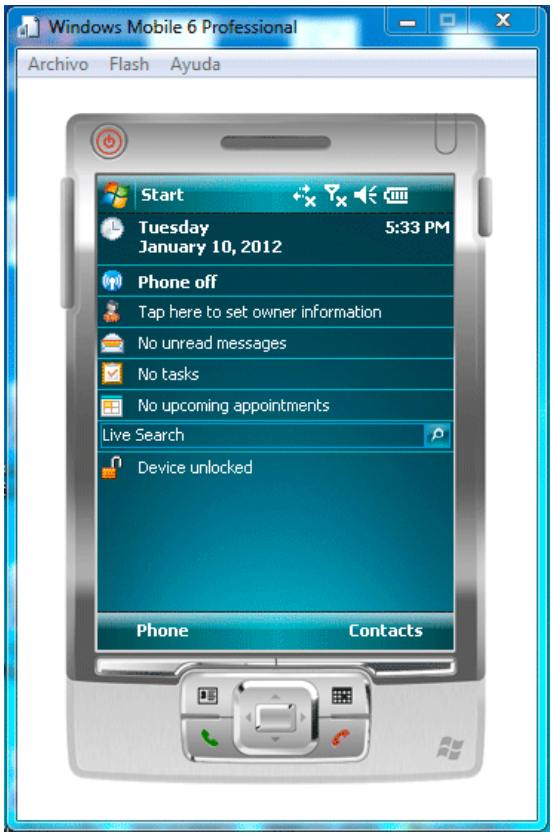

Figura 3: Emulador Windows Mobile

2.5 Microsoft Visual Studio 2008

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2008 fue publicado (RTM) el 17 de noviembre de 2007 en inglés, mientras que la versión en castellano no fue publicada hasta el 2 de febrero de 2008. La principal novedad es que venía integrado con el .Net Framework 3.5.

3. Herramientas: Desarrollo en Android

3.1 Introducción

Android ([6]) es un sistema operativo para dispositivos móviles como teléfonos inteligentes y tabletas. Es desarrollado por la Open Handset Alliance, la cual es liderada por Google.

Figura 4: Logo Android

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un Framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Además, existe una gran oferta de software de terceros disponible para Android, la cual se puede adquirir a través del Android Market, que es un catálogo oficial de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC, Markets de terceros como la App Store de Amazon, o simplemente descargando los archivos autoinstalables "`.apk`" en el dispositivo.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

3.2 Lenguaje de programación – Java

Java ([7][8]) es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Las aplicaciones Java están típicamente compiladas en un bytecode (aunque la compilación en código máquina nativo también es posible) lo que permite se ejecución en cualquier sistema operativo con una maquina virtual java.

A continuación se puede observar el típico ejemplo de "Hola Mundo", que como podemos observar es muy parecido al usado en C#:

```
public class HelloWorld
{
  public static void main(String args[])
  {
     System.out.println("Hello, World!");
  }
}
```

El término Java no tiene un origen conocido, aunque la hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de café caliente.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

3.3 Android SDK

El SDK de Android ([9]) proporciona las herramientas y las API necesarias para empezar a desarrollar aplicaciones que se puedan ejecutar en dispositivos con la tecnología de Android.

El SDK nos ofrece, además de un emulador, todas las aplicaciones y librerías que vamos a necesitar para desarrollar aplicaciones. El emulador recrea perfectamente un móvil Android, por lo que, en principio, casi todo el desarrollo podremos hacerlo sin usar un terminal real. En la Figura 5 se puede observar un ejemplo de emulador.

Figura 5: Emulador Android

Podemos programar con el SDK directamente, aun que resulta bastante engorroso, Afortunadamente, han pensado en todo y han creado un plugin para Eclipse que nos va a facilitar mucho el trabajo (Ver anexo instalación ADT de eclipse).

3.4 Eclipse

Eclipse ([10]) es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en

red como Telnet y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos plugins estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados como PHP o Javascript.

Mediante el plug-in de Android ([11]), Eclipse no solo permite programar aplicaciones para el sistema operativo Android, permite actualizar la SDK, configurar emuladores y ejecutar en modo depuración cualquier aplicación ya sea sobre el emulador o un dispositivo Android.

4. Diseño e implementación del marco genérico

Una vez se han analizado las tecnologías a utilizar en el proyecto, se diseñará el marco genérico valido para las diferentes plataformas móviles, para luego proceder al desarrollo de una aplicación basada en este marco demostrando como que se puede utilizar y su utilidad.

4.1 Requisitos

Los sistemas operativos móviles son muy diversos, de cara a la implementación de aplicaciones, deberemos especificar unos requisitos que las plataformas deberán cumplir para poder implementar aplicaciones con este marco:

- Independencia del lenguaje de programación.
- Pantalla para presentar los datos.
- Un sistema de interacción con el usuario: Pantalla táctil.
- Basado en un lenguaje que permita pintar a bajo nivel.
- Conectividad (No todas las aplicaciones la requieren)

En los siguientes puntos explicaremos detalladamente la importancia de cada uno de estos puntos de cara a nuestro diseño.

4.2 Diseño

A continuación se hará un análisis de los aspectos más relevantes a la hora de pensar en el diseño de esta aplicación en función de los requisitos y necesidades descritos en el apartado anterior.

El esquema general de uso de las aplicaciones marco genérico es el siguiente:

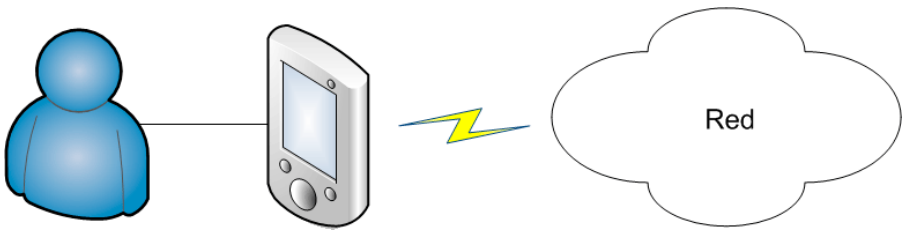

Figura 6: Esquema general del marco

4.2.1 Casos de Uso

En ingeniería del software, se llama caso de uso a la técnica para encontrar los requisitos posibles de un sistema nuevo o una actualización de software. Con la intención de conseguir un objetivo específico, cada caso de uso proporcionará un escenario que indica cómo debería interactuar el sistema con el usuario. Estos diagramas son muy útiles para especificar el comportamiento de un sistema con los usuarios u otros sistemas, es decir, muestran las relaciones entre los actores que intervienen los casos de uso en un sistema.

Los principales casos de uso que tendrá el marco genérico serán los siguientes:

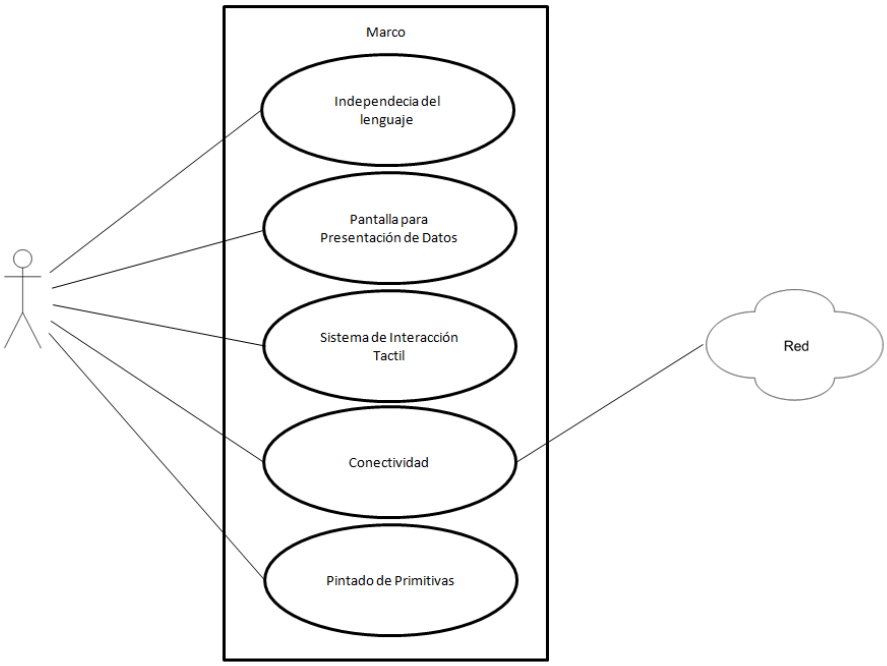

Figura 7: Casos de Uso

4.2.2 Independencia del lenguaje

Para que el marco genérico se pueda adaptar bien a cualquier sistema operativo móvil, se debe intentar diseñar sin tener en cuenta las cualidades de los lenguajes de programación propios de cada sistema operativo móvil, de forma que la tarea de migración se vea simplificada.

4.2.3 Pantalla de presentación de datos

La pantalla sirve para presentar la información a los usuarios de forma visual. En ella se muestra la información que se pretende mostrar al usuario. En los dispositivos móviles las pantallas suelen tener un tamaño reducido, lo que implica que la información que se puede

mostrar en una pantalla es limitada, por lo el marco debe tener mecanismos para maximizar la información que se muestra en la pantalla.

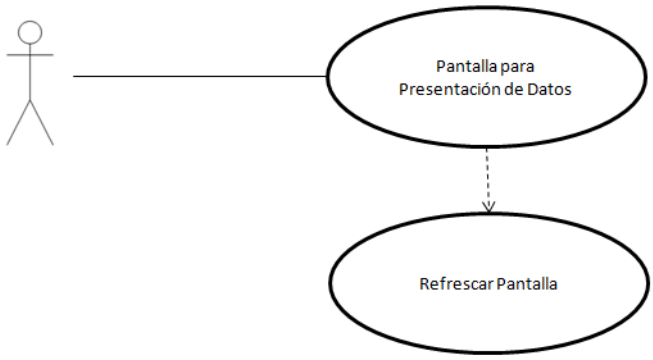

Figura 8: Casos de Uso presentación de datos

4.2.3.1 Refrescar Pantalla

Como se ha comentando en el apartado anterior la información se debe poder mostrar al usuario utilizando la pantalla, para ello se usará un temporizador del sistema que se encargará de repintar la información en pantalla a un velocidad que permita hacer que la aplicación sea usable y a su vez no malgaste procesador.

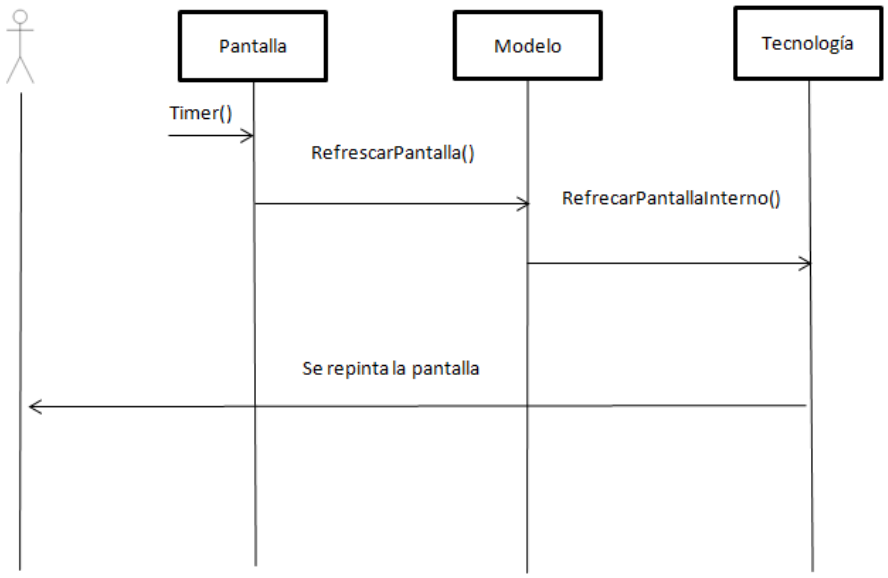

Figura 9: Diagrama Secuencia Refrescar Pantalla

En la Figura 10 podemos observar con algo más de detalle como funcionaría este proceso usando un tiempo de refresco de 50 ms:

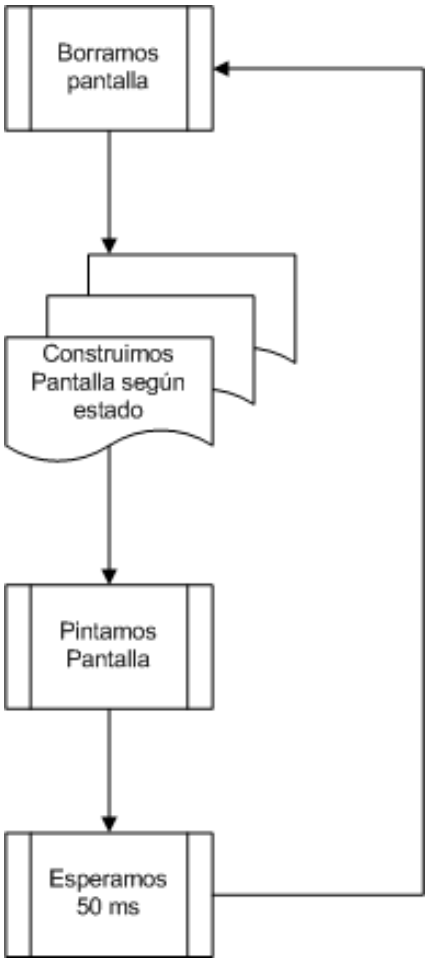

Figura 10: Diagrama Estados Refrescar Pantalla

Como se puede observar, lo primero que se hace es borrar la pantalla, después se mira el estado de la aplicación y se pinta la pantalla concreta con todas sus componentes, este proceso se repite cada 50 ms.

Para mejorar el comportamiento de este proceso y evitar el efecto de *aliasing*, las componentes de una pantalla se pintan en una imagen antes de pintarse en la pantalla, y luego esta imagen completa es la que se lleva a la pantalla.

El ojo humano es capaz de ver y distinguir imágenes a una velocidad 25 hz de media, lo que implica que los cambios de imagen por debajo de los 40 ms deberían ser imperceptibles. En la aplicación usando el proceso de *anti-aliasing* no necesitamos refrescar la pantalla a esa velocidad para que el comportamiento sea bueno, pero usa 50 ms (tiempo ligeramente mayor) para que la acción sobre las componentes de la pantalla sea realista, por ejemplo, al pulsar un botón, desde que se pulsa hasta que se ve pulsado pasará como máximo este tiempo de refresco, si coincidiese que justo se ha pulsado el botón al inicio del tiempo de refresco.

4.2.4 Sistema de interacción táctil

Cualquier aplicación debe permitir al usuario interactuar con ella. En los dispositivos móviles de última generación nos encontramos que las pantallas son táctiles, además de incluir algunos botones hardware. Estos botones son diferentes para cada plataforma, mientras que la pantalla es algo común a todos.

Para poder utilizar el marco genérico el sistema operativo móvil tiene como requisito detectar tres tipos de eventos:

- Pulsar Pantalla
- Mover el dedo/puntero tras pulsar la pantalla
- Levantar el dedo/puntero de la pantalla

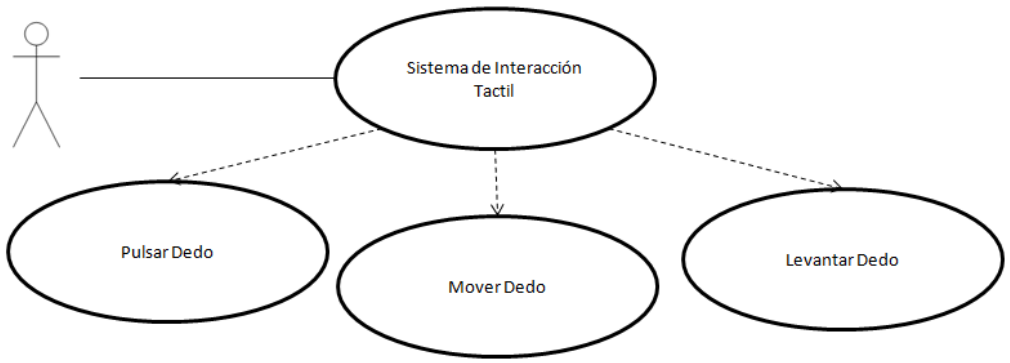

Figura 11: Casos de Uso del Sistema de interacción táctil

Con estos tres casos de uso podemos realizar la interacción típica de cualquier aplicación, desde pulsar botones, scroll(deslizado) de información en la pantalla o por ejemplo hacer swipe para pasar entre pantallas (movimiento rápido del dedo de derecha a izquierda o viceversa).

4.2.4.1 Pulsar, Mover y Levantar Dedo

Como se ha comentado en el apartado anterior necesitamos modelar los tres eventos que se producen con la pantalla táctil. Estos tres eventos tienen la cualidad de que se producen de manera ordenada:

1- Evento de pulsar pantalla

2- Número indefinido de eventos moverse sobre la pantalla

3- Evento levantar dedo de la pantalla

En el diagrama de la Figura 12 podemos observar el diseño de estos tres casos de uso, suponiendo que se ha producido sólo un evento de movimiento.

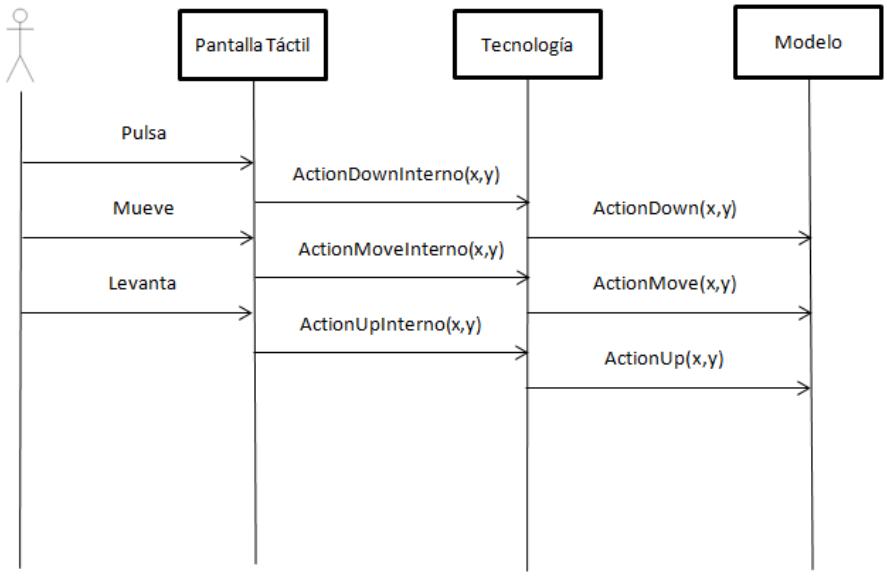

Figura 12: Diagrama Secuencia sistema de interacción táctil

En la Figura 12 tenemos representado al usuario, la pantalla táctil del terminal, la tecnología interna del móvil y modelo de aplicación diseñado. Cuando el usuario pulsa la pantalla la combinación de hardware y sistema operativo generará un evento indicando a la aplicación dicha acción. Esto mismo ocurre si desplaza el dedo con la pantalla pulsada y para terminar cuando deja de pulsarla.

4.2.5 Conectividad

Las aplicaciones de forma general necesitan obtener información de la red para mostrarla a los usuarios. Debido a que la conectividad de los dispositivos es limitada y normalmente asociada a tarifas de datos, es importante crear un sistema que sea eficiente a la hora de descargar datos de la web.

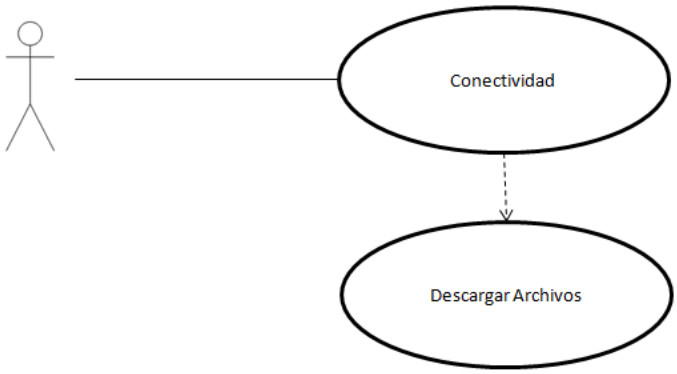

Figura 13: Casos de Uso Conectividad

Una decisión de diseño es que los archivos que obtenemos de la red los almacenamos en memoria. Esta decisión viene motivada por varias razones:

- Imágenes: No descargar varias veces la misma imagen, comprobamos primero si esta en memoria.
- Textos: Esperar a que el archivo este descargado, para comprobar su integridad antes de procesar los datos.
- Evitar errores por cortes de la conexión.

Normalmente estos archivos son alojados en dominios web (El servidor de datos usa un sistema ftp, por eficiencia y rapidez), ya que todas las plataformas móviles disponen de mecanismos para poder obtener datos mediante URLs.

4.2.5.1 Descarga de archivos

Como se ha especificado en el punto anterior se tiene que definir un sistema genérico de descarga de archivos que podamos implementar en las diferentes plataformas.

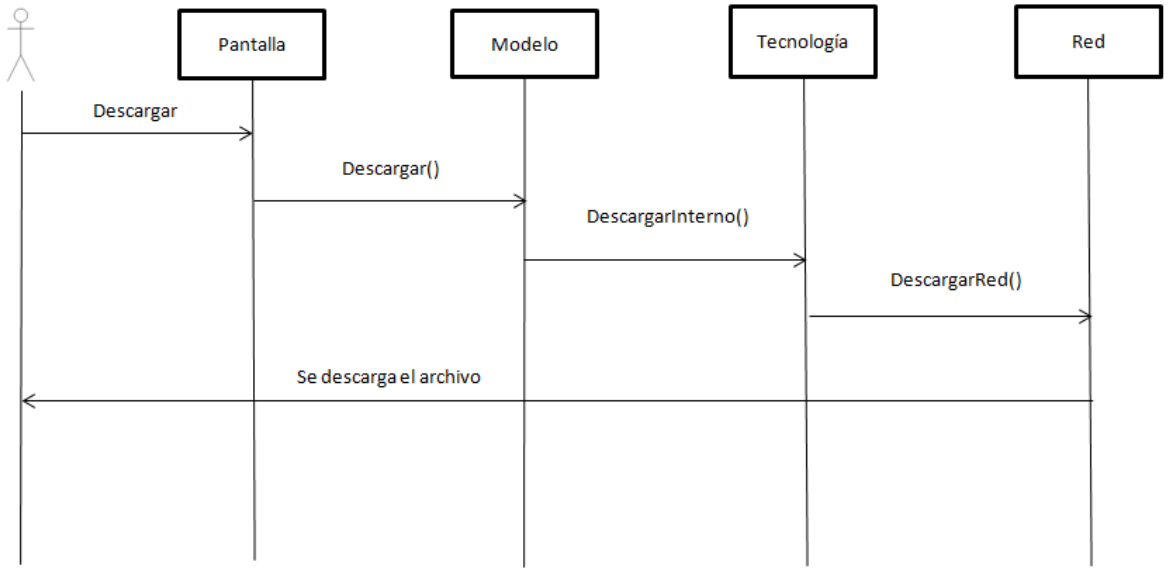

Figura 14: Diagrama de secuencia de descarga de archivos

En la Figura 15 podemos apreciar el desarrollo del proceso desde un punto de vista más funcional.

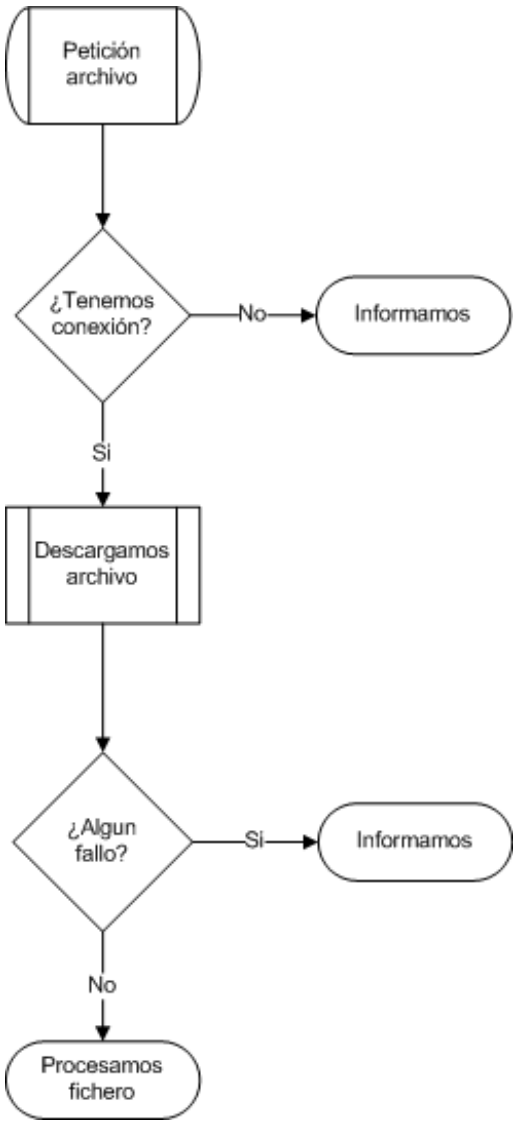

Figura 15: Diagrama de estados de descarga de archivos

Como podemos observar en la Figura 15 se debe comprobar el estado de la conexión antes de descargar el archivo y después del proceso de descarga comprobar que el archivo se ha descargado correctamente. Se debe tener en cuenta que según la zona y la cobertura, es muy probable que se pueda perder la conexión.

Si el archivo que se está bajando es crítico o importante para la aplicación, si se produce un fallo en la descarga, podremos realizar un número de reintentos controlado (número de iteraciones o tiempo) antes de informar del problema.

4.2.6 Pintado de Primitivas

Todos los lenguajes de programación se sirven de librerías y funciones de alto nivel que nos ayudan a programar la parte grafica de la aplicación. Cuando queremos utilizar un marco que nos permita migrar aplicaciones entre diferentes plataformas móviles nos encontramos con que este tipo de librerías o funciones son muy diferentes, por lo que nos resulta imposible utilizarlas y tenemos que utilizar funciones de bajo nivel, que son prácticamente iguales para todas las aplicaciones.

En concreto para usar este marco general de programación, las siguientes funciones son un requisito:

- Escribir un texto en pantalla con un tamaño especifico
- Dibujar una imagen en pantalla con tamaño especifico
- Dibujar una línea de un determinado color

La última no es obligatoria, puesto que usando imágenes se podría sustituir, pero el coste de memoria y carga de las pantallas se vería bastante mermado.

En la Figura 16 podemos observar el diagrama con los casos de uso de pintado de primitivas.

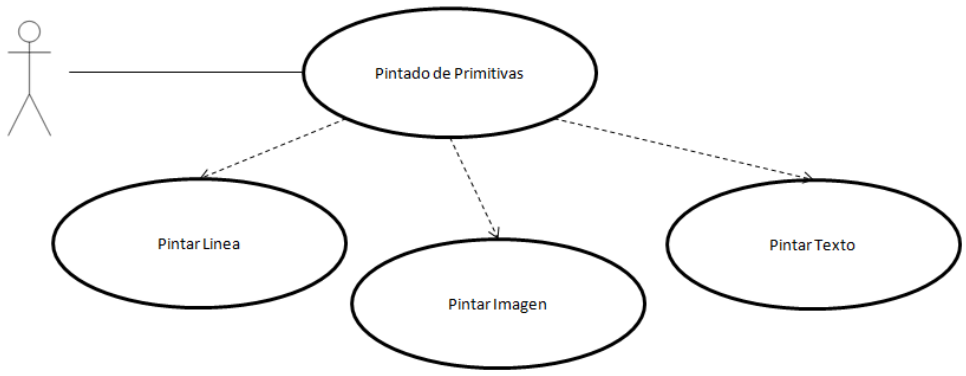

Figura 16: Casos de Uso de Pintando primitivas

Como se ha dicho anteriormente en este apartado, el uso de pintar línea no es necesario, pero si recomendable, como este caso, otros métodos prácticos y no han sido necesarios en este proyecto, podrían ser pintar eclipse, pintar rectángulo, pintar polígono, lo que significa el marco general es sólo una base que puede ir incrementándose continuamente en función de los requisitos de cada proyecto.

4.2.6.1 Pintar Línea

Se utilizará el método o métodos de la tecnología correspondiente destinado a esta función.

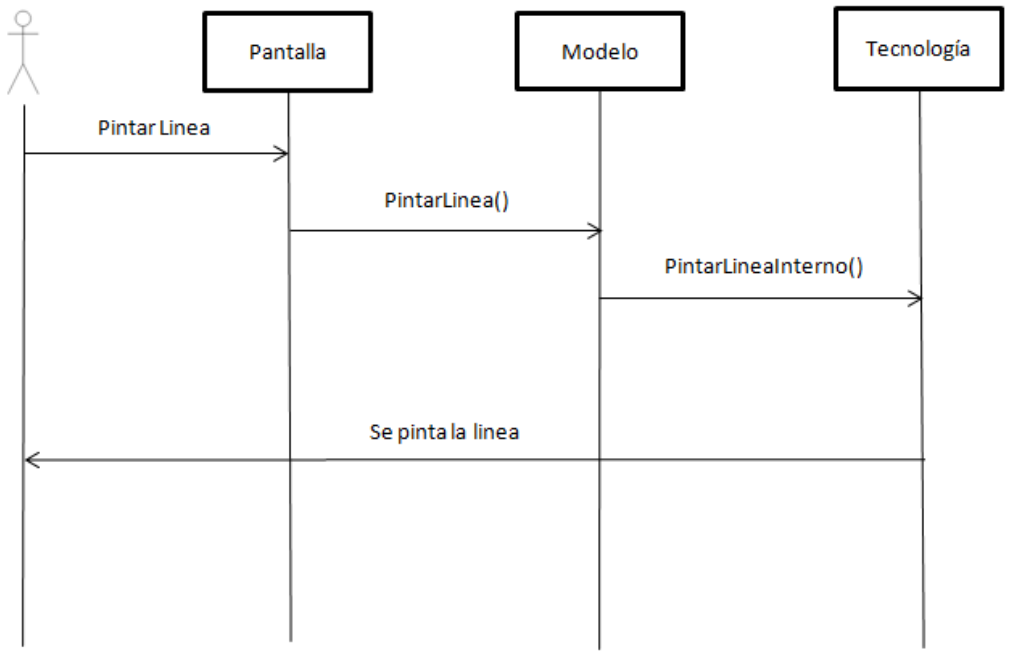

Figura 17: diagrama de secuencia de Pintar Línea

Como se puede observar en la Figura 17, cuando se requiera pintar una línea en la pantalla, se llamará a nuestro método genérico pintar línea, que como norma general se la pasarán como parámetros el punto de origen y de destino. Estos puntos tienen dos coordenadas, horizontal y vertical, puesto que las pantallas se construyen como imágenes de dos dimensiones para nuestras necesidades.

Al igual que se ha comentado en apartados anteriores, este es un marco base, y se podría indicar que se podría mejorar con pantallas que usaran renderizado 3D, pero la realidad es que sería un marco totalmente diferente que se basaría en las diferentes tecnologías que usan las plataformas móviles (OpenGL para Android, XNA para Windows Phone), y que no sería útil para la mayoría de aplicaciones del mercado actual, que es un mercado de aplicaciones con pantallas en 2D.

4.2.6.2 Pintar Imagen

Se utilizará el método o métodos de la tecnología correspondiente destinados a esta función.

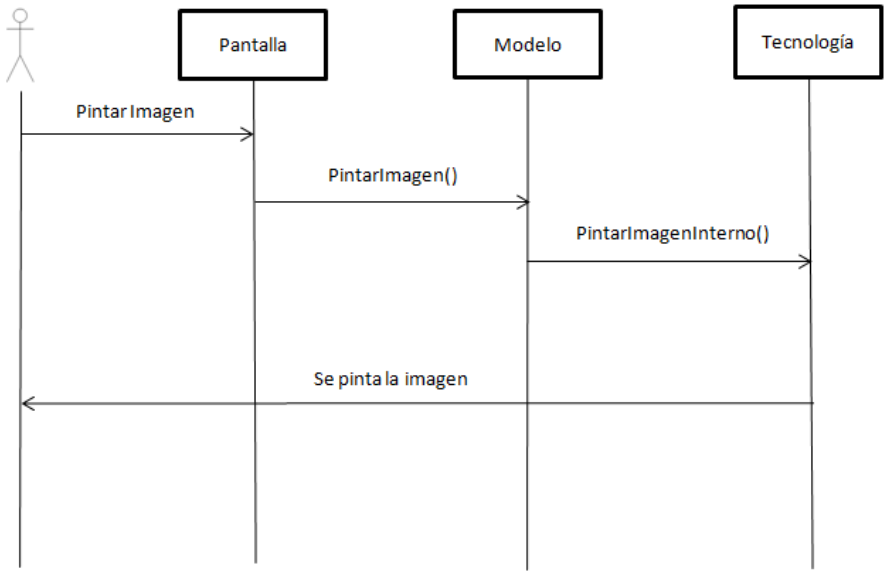

Figura 18: Diagrama de secuencia de Pintar Imagen

Como se puede observar en la Figura 18, cuando se requiera pintar una imagen en la pantalla, se llamará a nuestro método genérico pintar imagen, que como norma general se la pasarán como parámetros el punto de origen y el tamaño de la imagen. Usan do el tamaño de la imagen, los métodos internos del marco se encargaran de escalarla desde su tamaño original al tamaño deseado en pantalla.

Se debe tener en cuenta dos factores, si la imagen original es más pequeña que la que se desea pintar en pantalla, la imagen se verá pixelada, efecto no deseado de cara a un buen diseño de la aplicación, y si la imagen es más grande que la que se pinta en pantalla, el tamaño de memoria que se usa para cargarla será ineficiente. El problema es que estos dispositivos tienen muchos tamaños de pantalla diferente, y como norma general a menor pantalla peor dispositivo y menos memoria, lo que implica que deberemos jugar con ambos valores (permitir un poco de pixelación y un poco de ineficiencia en memoria) para que el resultado sea bueno en todo tipos de dispositivos.

Para evitar este problema el sistema operativo Android tiene diferentes carpetas para introducir las imágenes, de forma que para cada tipo de resolución use las de una carpeta o las de otra. En Windows Mobile esto no lo crea el sistema operativo, pero teniendo en cuenta que es un sistema poco nada restrictivo a la hora de acceder a los recursos en un sistema de ficheros, este mismo proceso de selección de imágenes en función de la resolución se podría implementar tras leer el alto y ancho de la pantalla nada más arrancar la aplicación. Este proceso que tienen Android y que podría implementarse en Windows Mobile es importante conocerlo, pero no se va a usa en este marco genérico, puesto que lo que se está buscando desde el principio es ahorrar tiempos de coste en la producción de las aplicaciones, y sacar las imágenes para cada resolución es un trabajo inicialmente costoso.

4.2.6.3 Pintar Texto

Se utilizará el método o métodos de la tecnología correspondiente destinados a esta función.

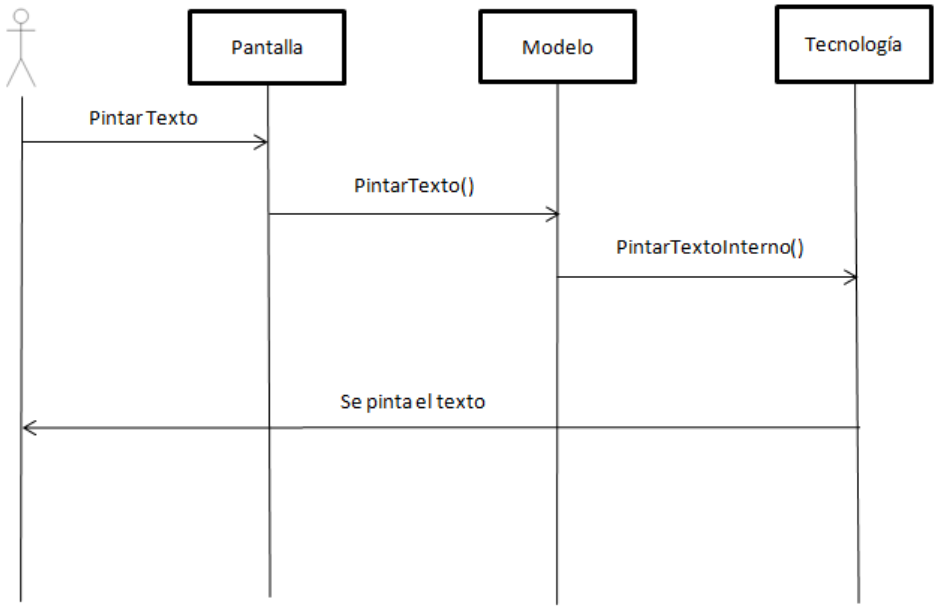

Figura 19: Diagrama de secuencia de Pintar Texto

Como se puede observar en la Figura 19, cuando se requiera pintar un texto en la pantalla, se llamará a nuestro método genérico pintar texto, que como norma general se la pasarán como parámetros el punto de origen, el tamaño de letra, el estilo de letra (normal o negrita) y la alineación del texto (derecha, centrado, o izquierda respecto el punto de origen).

En este marco base sólo usamos estilos de letra normal y negrita, pero si el proyecto lo viera necesario, se podría aumentar la funcionalidad del marco usando cursiva, subrayado, o cualquier otro estilo que los sistemas operativos nos permitiesen usar o crear (subrayado se podría crear como un texto normal y pintar una línea debajo del texto).

4.2.7 Otras consideraciones

Otras consideraciones que debemos tener en cuenta a la hora de realizar la mayoría de las aplicaciones, es elegir unos estándares que funcionen en la mayoría de las plataformas móviles. Por ese motivo se toman las siguientes consideraciones:

- Acceso datos red: Mediante peticiones Http tipo “GET”.
- Imágenes: Formato .png
- Datos: Archivos .xml

Estas consideraciones se han tenido en cuenta a la hora de decidir las plataformas Android y Windows Mobile para la realización del proyecto, pero son también las más obvias para trabajar otras plataformas móviles como iOS (iPhone e iPad), Blackberry o Windows Phone 7, puesto que todas estas plataformas soportan estas tecnologías debido a su amplia difusión.

4.2.8 Funcionamiento de las aplicaciones basadas en el marco genérico

El caso de uso general de la aplicación se puede mostrar en la siguiente imagen. Dependiendo de la aplicación, este diagrama se completará con las funciones pertinentes.

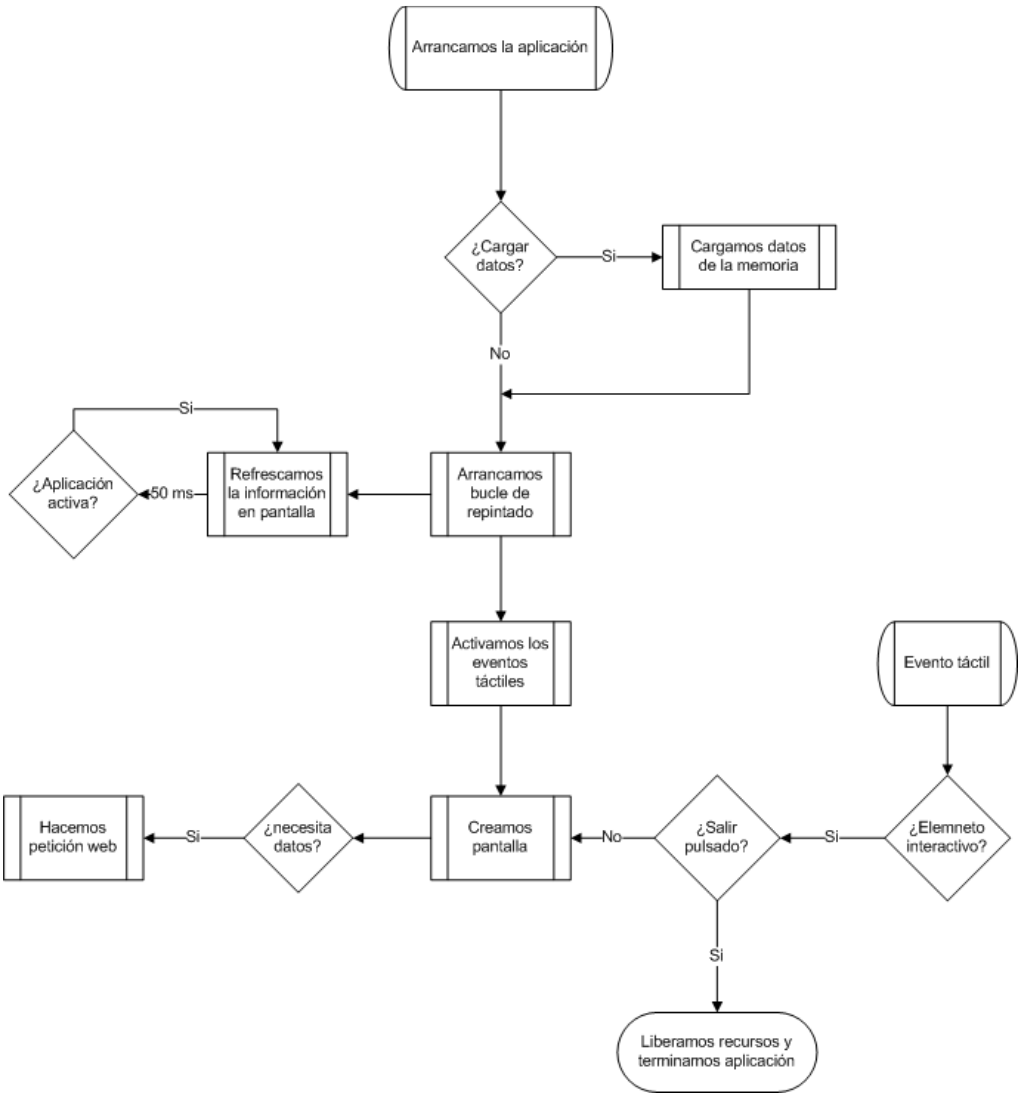

Figura 20: Diagrama de estados de aplicación básica

Como podemos observar en la Figura 20, cuando se arranca la aplicación se cargan los datos que se desean usar/mostrar si es necesario, y ponemos en marcha el sistema de repintado de la pantalla en función del estado de la aplicación.

En las aplicaciones que haya que cargar muchos datos al arrancar, como es el caso de la aplicación ejemplo que se va a desarrollar en este proyecto (Mundial de fútbol 2010), se hace una carga inicial de datos con los recursos indispensables, se inicia el bucle de repintado, y se pone la aplicación en un estado de carga, que pintará una pantalla mientras se carga el resto de información, no dando al usuario la sensación de que la aplicación está bloqueada y que se el tiempo que este activa dependerá del sistema operativo, la memoria y el procesador del dispositivo, pudiendo ser desde unos pocos milisegundos en dispositivos modernos de gama alta a varios segundos en dispositivos viejos de gama baja (los dispositivos modernos de gama baja ya disponen de procesadores muy rápidos).

El refresco de la pantalla se ha tratado anteriormente con detalle, pero es importante reflejar que este proceso funciona como un hilo independiente de la aplicación que estará activo hasta que esta se cierre. Como hilo independiente a la hora de utilizarlo se le puede asignar una prioridad, de forma que tenga más o menos que el resto de hilos de la aplicación. Se ha comprobado que con la prioridad estándar el comportamiento de pintado de pantalla es suficientemente bueno y no es necesario aumentar su prioridad.

Mediante los eventos táctiles se detectara la interacción del usuario con la aplicación, si este desea realizar un *scroll* de la pantalla, pulsar un botón, hacer *swipe*, o cualquier otra operación que se nos ocurra programar con los métodos que nos proporciona el marco (por ejemplo que al hacer una diagonal con el dedo en la pantalla se vaya a la pantalla home).

Cuando se sale de la aplicación se deben liberar todos los recursos usados en la aplicación y parar todos los hilos. Actualmente hay muchos sistemas operativos móviles o nuevas versiones de estos, que se encargan de esta tarea aunque el desarrollador no lo haya hecho correctamente o de manera intencionada, pero en Windows Mobile y en las primeras versiones de Android, si se dejaba un hilo funcionando en segundo plano tras cerrar la aplicación, este continuaba con su tarea. Como se ha dicho, esto puede ser algo intencionado, por ejemplo en aplicaciones de seguimiento de flotas, donde se desea enviar la posición de un vehículo cada cierto tiempo a un servidor, pero existen herramientas mejores y más especificas para realizar este tipo de tareas.

4.2.9 Diseño de componentes de alto nivel

En las diferentes tecnologías existen componentes de alto nivel creadas para facilitar el desarrollo de aplicaciones, pero tienen la desventaja de que su utilización suele ser muy diferente según que tecnologías. Esto no supone un problema, puesto que con los casos de uso que hemos definido podemos crear las componentes de alto nivel que podamos necesitar.

Por ejemplo, en Android tenemos la clase *android.widget.Button*, y en Windows Mobile la clase *System.Windows.Forms.Button*, pero ambas se añaden a la pantalla de forma diferente, y se configuran con diferente código, por lo que no nos permiten migrar el código de manera sencilla. Pero un botón, en el fondo, son dos imágenes (pulsadas y no pulsadas) que varían su comportamiento en función de la interacción del usuario. Entonces se puede crear nuestra propia clase botón que utilizando primitivas de dibujado tenga dos imágenes, y que implemente los tres métodos táctiles para decidir qué imagen se pinta, y que se puede utilizar en diferentes plataformas móviles sin necesidad de utilizar las librerías de alto nivel de cada sistema.

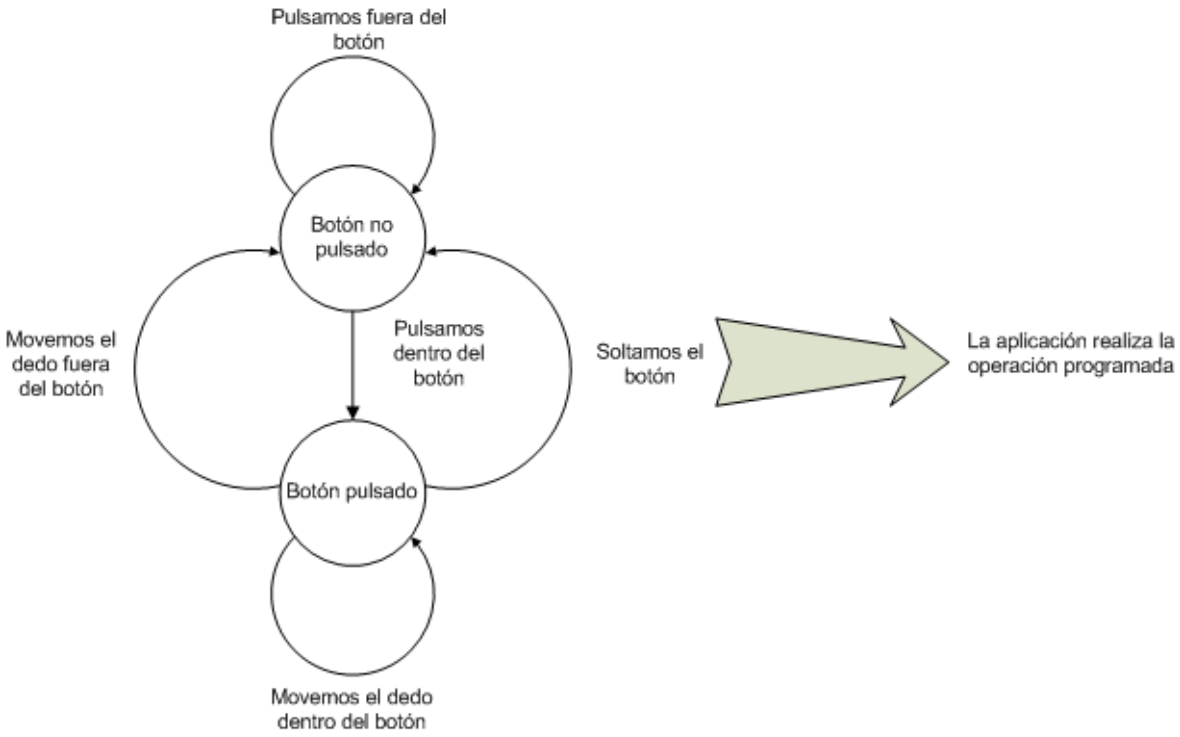

Figura 21: Diagrama de estados de un botón

Como se puede ver en la Figura 21 el botón tiene dos estados, pulsado y no pulsado, de forma que se pintará la imagen correspondiente en función del estado, provocando que el usuario pueda ver la interacción con la componente. Cuando se suelta el botón se lanza el evento correspondiente para realizar la tarea que este programada.

4.3 Mapeo en Windows Mobile

La implementación del marco se va a centrar en las dos plataformas que dominaban el mercado de los smartphones en el 2010, Windows Mobile y Android.

Para el mapeo en Windows Mobile se utilizará el lenguaje de programación C#, la herramienta de desarrollo Visual Studio 2008 Profesional con las librerías de Windows Mobile 6.0 y 6.5 instaladas. Para las pruebas iniciales se usará el emulador Windows Mobile Classic.

4.3.1 Refrescar Pantalla

Para la implementación de refresco de pantalla tenemos en la librería System.Windows.Forms de Windows Mobile el método Invalidate.

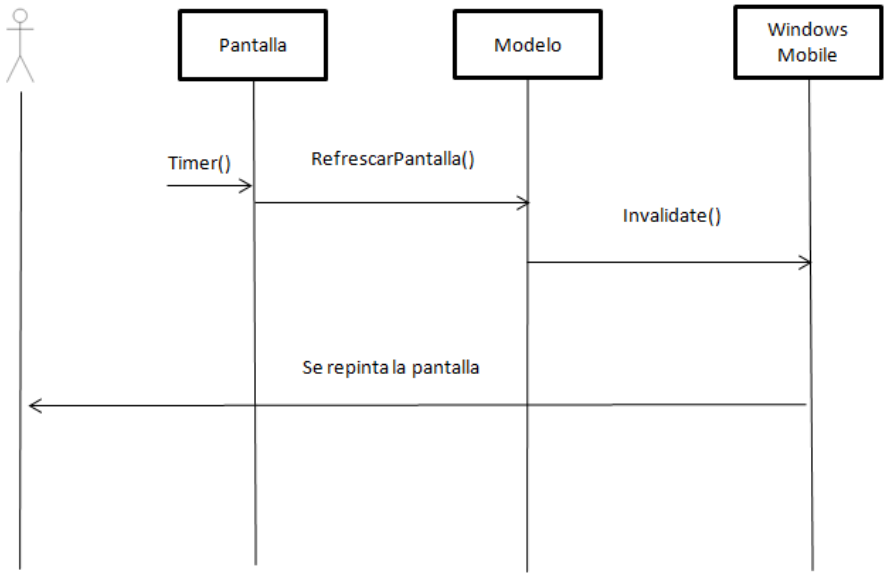

Figura 22: Diagrama de implementación de refrescar pantalla Windows Mobile

4.3.2 Pulsar, Mover y Levantar Dedo

Para la implementación de la interacción táctil tenemos en la librería System.Windows.Forms de Windows Mobile tres eventos diferenciados, que podemos asociar a cada uno de los métodos del marco genérico.

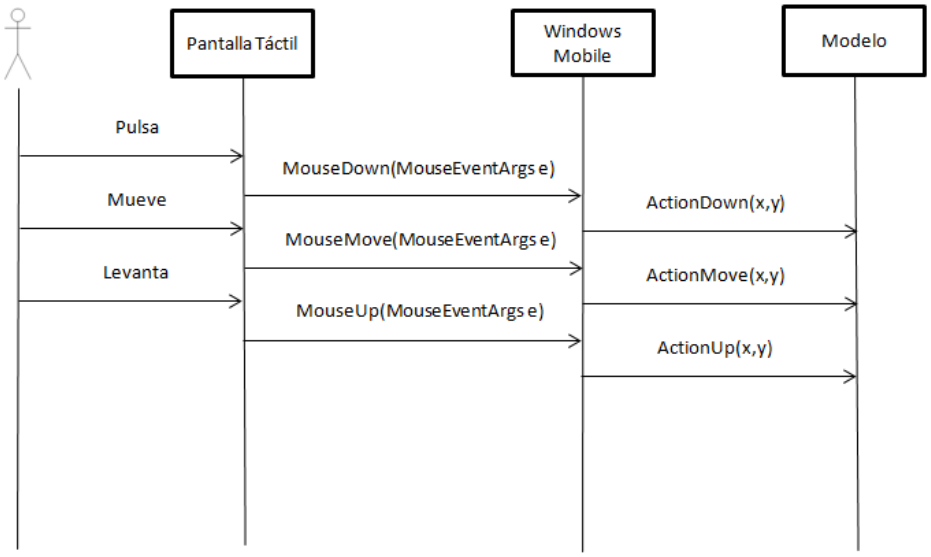

Figura 23: Diagrama de implementación de eventos táctiles Windows Mobile

4.3.3 Descarga de archivos

Para la implementación de la descarga de archivos utilizaremos varios métodos de las librerías System.IO y Sytem.Net de Windows Mobile

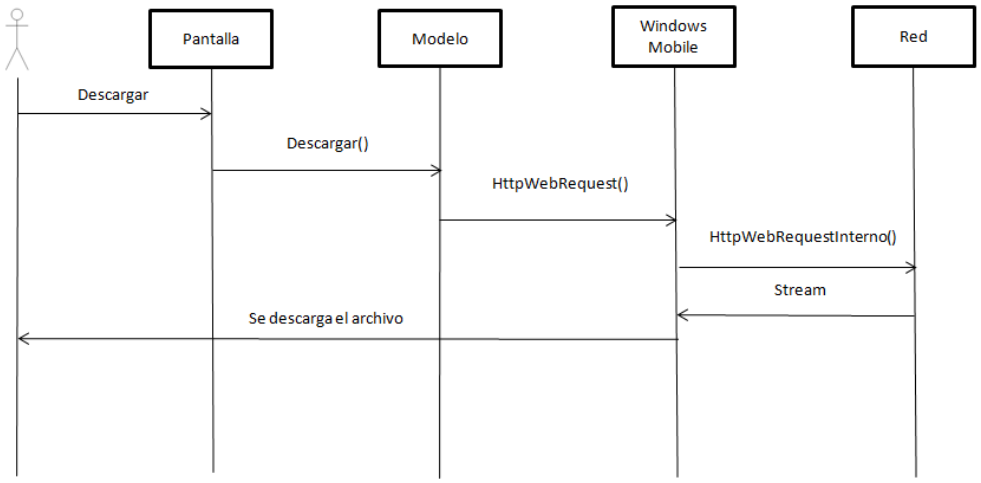

Figura 24: Diagrama de implementación descargar archivo Windows Mobile

4.3.4 Pintar Línea

Para la implementación de refresco de pantalla tenemos en la librería System.Drawing de Windows Mobile el método DrawLine.

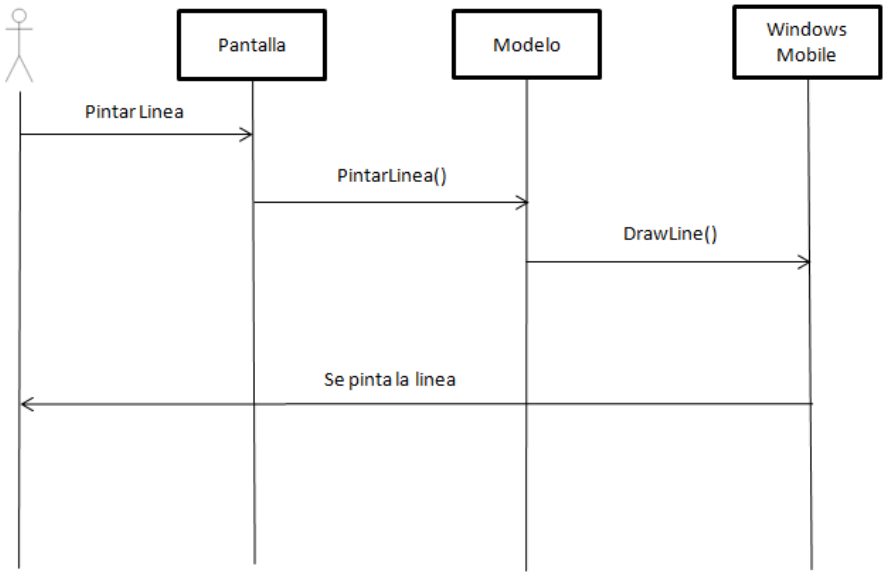

Figura 25: Diagrama de implementación de pintar línea Windows Mobile

4.3.5 Pintar Imagen

Para la implementación de refresco de pantalla tenemos en la librería System.Drawing de Windows Mobile el método DrawImage.

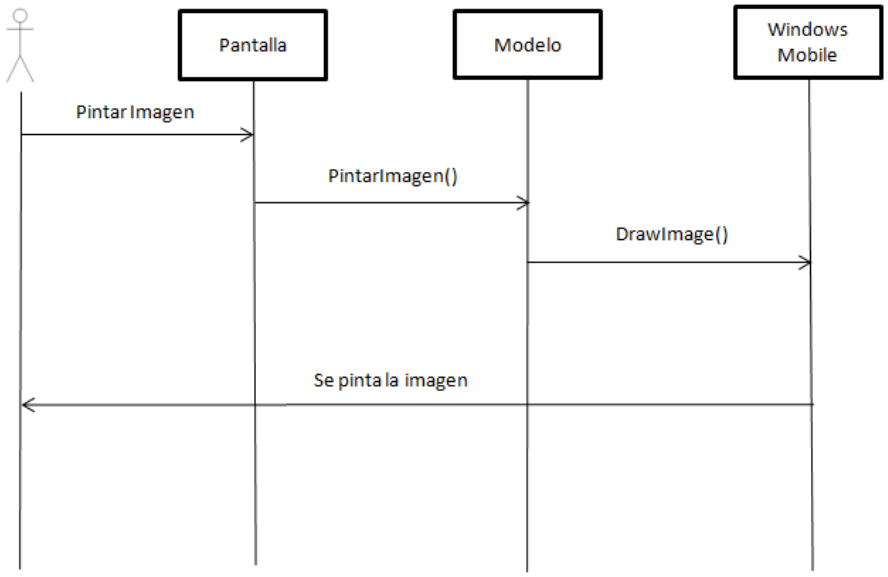

Figura 26: Diagrama de implementación de pintar imagen Windows Mobile

4.3.6 Pintar Texto

Para la implementación de refresco de pantalla tenemos en la librería System.Drawing de Windows Mobile el método DrawString.

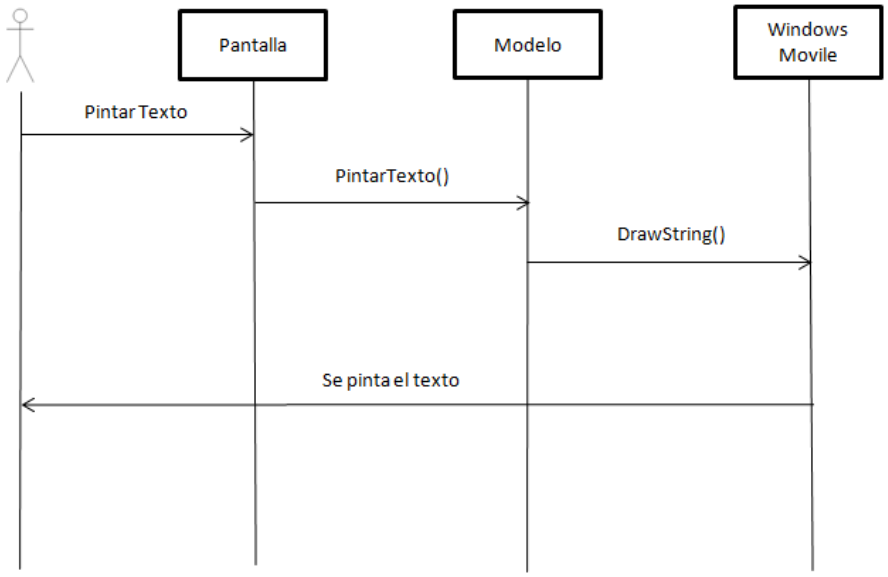

Figura 27: Diagrama de implementación de pintar texto Windows Mobile

4.4 Mapeo en Android

La implementación del marco se va a centrar en las dos plataformas que dominaban el mercado de los smartphones en el 2010, Windows Mobile y Android.

Para el mapeo en Android se utilizará el lenguaje de programación Java, la herramienta de desarrollo Eclipse con el ADT Plugin de Android instalado. Para las pruebas iniciales se usará el emulador QVGA con Android 1.6.

Se van a analizar los mismos casos que en Windows Mobile, de esta forma se dispondrá del mapeo de los métodos principales en ambas plataformas necesarios para el uso del marco genérico.

4.4.1 Refrescar Pantalla

Para la implementación de refresco de pantalla tenemos en la librería android.view.View de Android el método postInvalidate.

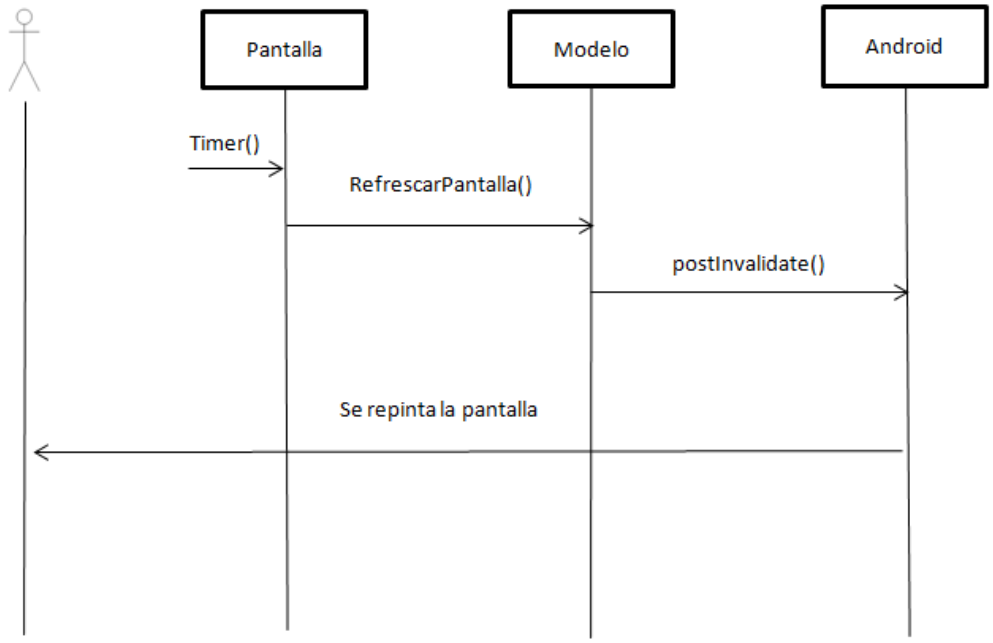

Figura 28: Diagrama de implementación de refrescar pantalla Android

4.4.2 Pulsar, Mover y Levantar Dedo

Para la implementación de la interacción táctil tenemos en la librería android.view.View de Android un sólo evento, que podemos asociar a cada uno de los métodos del marco genérico a través del valor de uno de los atributos que recibe el método.

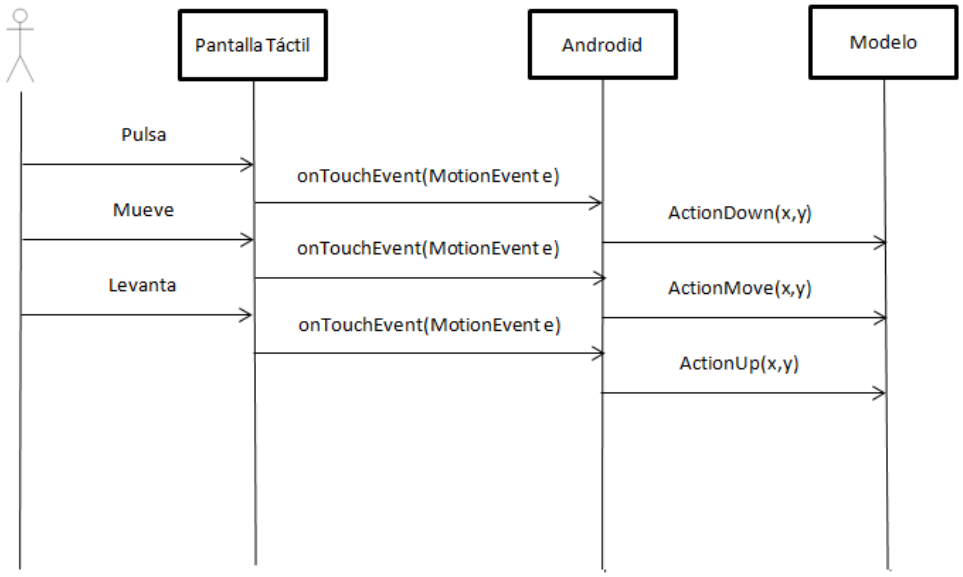

Figura 29: Diagrama de implementación de eventos táctiles Android

4.4.3 Descarga de archivos

Para la implementación de la descarga de archivos utilizaremos varios métodos de las librerías java.io y java.net de Android

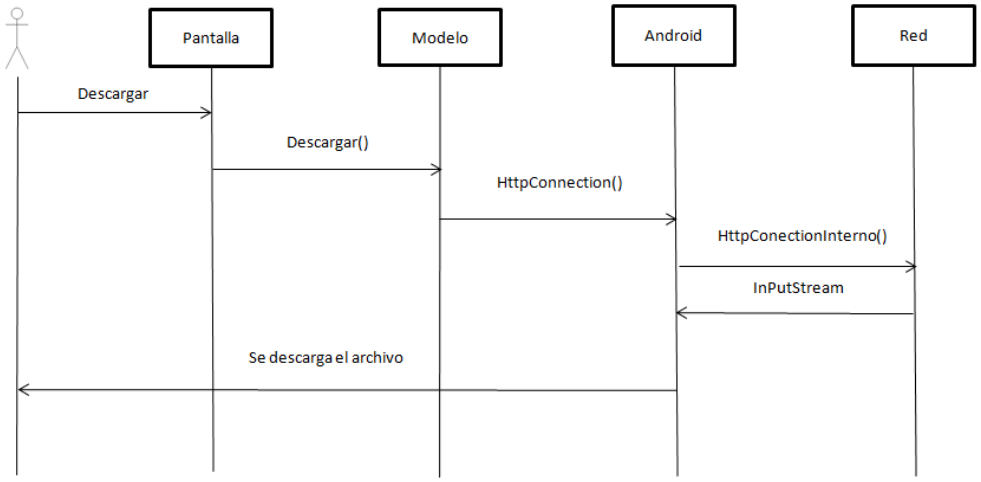

Figura 30: Diagrama de implementación descargar archivo Android

4.4.4 Pintar Línea

Para la implementación de refresco de pantalla tenemos en la librería android.graphics de Android el método drawLine.

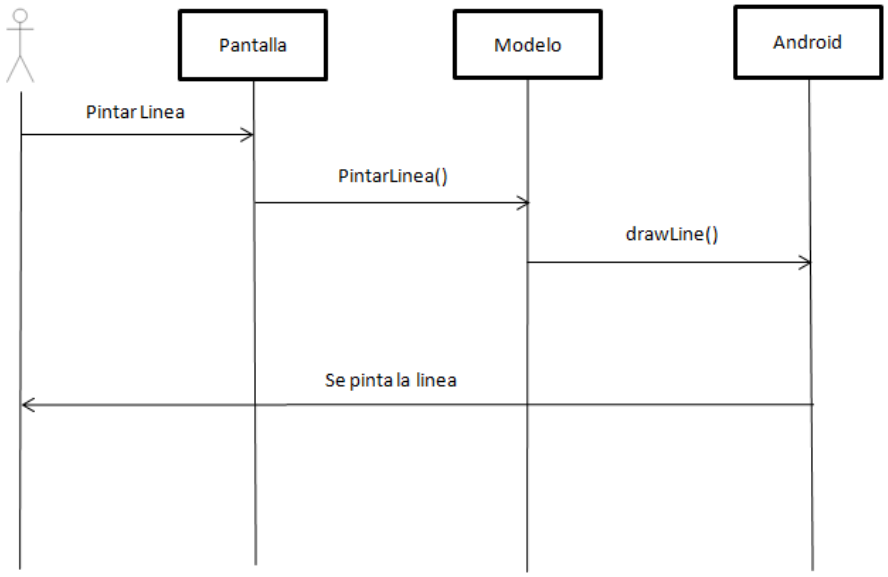

Figura 31: Diagrama de implementación de pintar línea Android

4.4.5 Pintar Imagen

Para la implementación de refresco de pantalla tenemos en la librería android.graphics de Android el método drawBitmap.

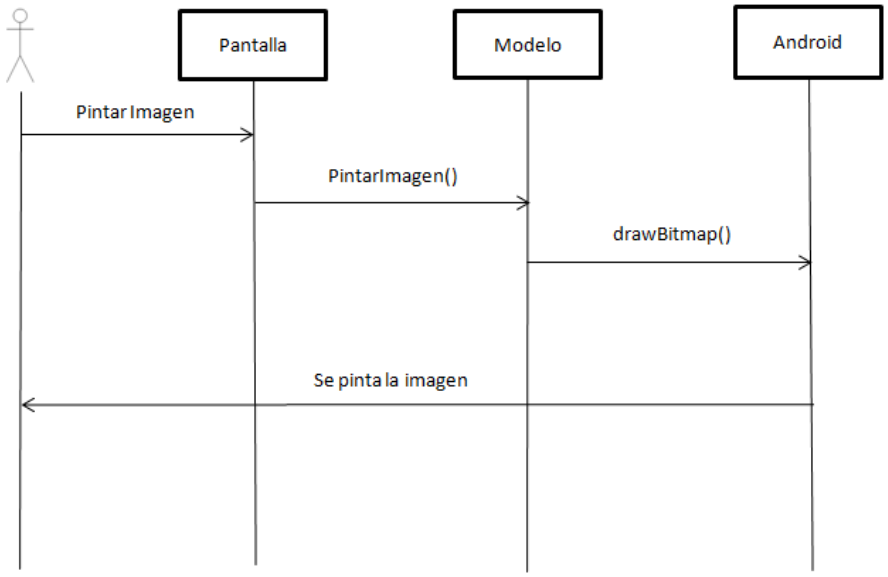

Figura 32: Diagrama de implementación de pintar imagen Android

4.4.6 Pintar Texto

Para la implementación de refresco de pantalla tenemos en la librería android.graphics de Android el método drawText.

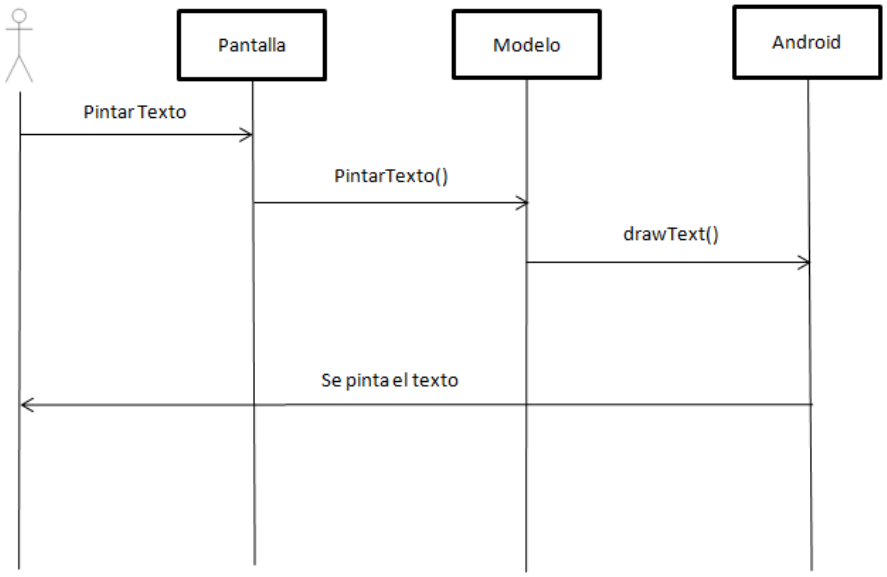

Figura 33: Diagrama de implementación de pintar texto Android

4.5 Reglas de transformación

Para poder migrar la aplicación entre dos lenguajes debemos definir unas reglas de transformación, algunas de ellas vienen marcadas por el diseño en sí del marco genérico y otras por los lenguajes de programación utilizados.

En nuestro caso he definido unas reglas para transformar el lenguaje C# en java. Estas reglas se pueden aplicar de forma bidireccional, es decir, no es necesario implementar siempre primero la aplicación en Windows Mobile y después migrarla a Android, sino que podemos realizar el proceso a la inversa. En la Tabla 1 podemos observar las reglas definidas.

Regla	Descripción
0	Reglas del Marco
1	Cabeceras de librerías
2	Namespace/Package
3	Listas y Arrays
4	Equals/equals
5	DateTime/Date
6	Strings

7	Int/Integer
8	Timers
9	Streams
10	XMLs
11	Mensajes
12	Exit
13	Multimedia

Tabla 1. Reglas de transformación

4.5.1 Regla 0: Reglas del Marco

Son las reglas derivadas de los mapeos de los casos de uso del marco genérico se ha definido, estas reglas son las básicas a aplicar, y el resto de ellas son un complemento basado en el estilo del lenguaje o en las librerías opcionales para la aplicación (mensajes de sistema, uso de fechas).

4.5.1.1 Pintar Línea

En los diferentes lenguajes de programación existe un objeto contenedor donde se puede pintar y que representa la pantalla, en Windows Mobile es un System.Drawing.Graphics y en Android es un android.graphics.Canvas. Este objeto tiene muchos métodos de pintado entre los que se encuentra poder pintar una línea.

Un ejemplo sería:

```
lapiz = new                    ⇔    lapiz = new Paint();
Pen(Color.Black,width)              lapiz.setColor(Color.BLACK);
g.DrawLine(lapiz, posX,             lapiz.setStrokeWidth(1);
posY, posX + lenX, posY);           g.drawLine(posX, posY, posX + lenX,
                                    posY, lapiz);
```

Para pintar las líneas podemos usar la transformación directa que vemos aquí arriba, o podemos usar el método auxiliar que hemos creado para tal efecto.

En Windows Mobile:

```
public static void pintarLinea(Graphics g, float posX1, float posY1,
float posX2, float posY2, float width, Color color)
{
     Pen lapiz = new Pen(color, Convert.ToInt32(width));
     g.DrawLine(lapiz, Convert.ToInt32(posX1), Convert.ToInt32(posY1),
                Convert.ToInt32(posX2), Convert.ToInt32(posY2));
}
```

En Android:

```
public static void pintarLinea(Canvas g, float posX1,float posY1, float
posX2, float posY2, float width, int color) {
      Paint lapiz=new Paint();
      lapiz.setColor(color);
      lapiz.setStrokeWidth(width);
      lapiz.setAntiAlias(true);
      g.drawLine(posX1, posY1, posX2, posY2, lapiz);
}
```

En ambos casos el resultado es el mismo, en el segundo el código es más parecido entre ambas plataformas, en el primero el código es más eficiente, pues podemos reutilizar la variable lápiz para pintar varias líneas.

4.5.1.2 Pintar Texto

En los diferentes lenguajes de programación existe un objeto contenedor donde se puede pintar y que representa la pantalla, en Windows Mobile es un System.Drawing.Graphics y en Android es un android.graphics.Canvas. Este objeto tiene muchos métodos de pintado entre los que se encuentra poder pintar un texto.

Un ejemplo sería:

```
sfCentrado = new                    ⇔     sfCentrado = new Paint();
StringFormat();                           g.drawText(textoC, lenX / 2, y + 16,
g.DrawString(textoC, font,                sfCentrado);
brocha, lenX/2, y,
sfCentrado);
```

Para pintar los textos podemos usar la transformación directa que vemos aquí arriba, o podemos usar el método auxiliar que hemos creado para tal efecto.

En Windows Mobile:

```
public const int ALING_CENTER = 0;
public const int ALING_LEFT = 1;
public const int ALING_RIGHT = 2;

public static void pintarTexto(Graphics g, String texto, float posX,
float posY, int size,Color color, int aling)
{
   Font font = new Font(FontFamily.GenericSansSerif, size,
                        FontStyle.Regular);
   StringFormat sf = new StringFormat();
   switch (aling)
   {
      case ALING_CENTER:
         sf.Alignment = StringAlignment.Center;
         break;
      case ALING_LEFT:
         sf.Alignment = StringAlignment.Near;
         break;
      case ALING_RIGHT:
         sf.Alignment = StringAlignment.Far;
         break;
   }
   SolidBrush brocha = new SolidBrush(color);
```

```
    g.DrawString(texto, font, brocha, Convert.ToInt32(posX),
                 Convert.ToInt32(posY), sf);
}
```

En Android:

```
public static final int ALING_CENTER=0;
public static final int ALING_LEFT=1;
public static final int ALING_RIGHT=2;

public static void pintarTexto(Canvas g, String texto, float posX, float
posY, int size,int color, int aling){
       Paint sf = new Paint();
       sf.setTypeface(Typeface.SANS_SERIF);
       sf.setTextSize(size*1.8f);
       switch (aling)
        {
             case ALING_CENTER:
                 sf.setTextAlign(Align.CENTER);
                 break;
             case ALING_LEFT:
              sf.setTextAlign(Align.LEFT);
                 break;
             case ALING_RIGHT:
              sf.setTextAlign(Align.RIGHT);
                 break;
        }
       sf.setColor(color);
       sf.setAntiAlias(true);
       g.drawText(texto, posX,posY+(sf.getTextSize()*0.9f), sf);
}
```

En ambos casos el resultado es el mismo, en el segundo el código es más parecido entre ambas plataformas, en el primero el código es más eficiente, pues podemos reutilizar varias variables, o incluso preparar una para cada alineación y usarla en toda la aplicación.

Aquí se pueden observar varias cosas importantes:

- En Android aunque se puede definir estilo negrita en el typeface, daba problemas en algunos dispositivos en sus primeras versiones, por lo que no vamos a usar negrita. Actualmente este problema esta corregido.
- La posición como se pinta el texto respecto unas coordenadas origen x e y, no es igual en ambas plataformas, por lo que se debe meter un factor de corrección (manualmente en el primer ejemplo, automáticamente basado en el tamaño del texto en el método que se ha creado).
- Los tamaños de letra y las fuentes son distintas para ambas plataformas. Debido a que el uso de una fuente no estándar es muy costoso para el procesador, se ha optado por usar estas en ambas plataformas. Además para que el tamaño en pantalla sea aproximadamente el mismo se aplica un factor de corrección sobre el tamaño del texto en el método creado para la plataforma Android.

4.5.1.3 Pintar Imagen

En los diferentes lenguajes de programación existe un objeto contenedor donde se puede pintar y que representa la pantalla, en Windows Mobile es un System.Drawing.Graphics y en Android es un android.graphics.Canvas. Este objeto tiene muchos métodos de pintado entre los que se encuentra poder pintar una imagen.

Un ejemplo sería:

```
Image bandera;
new Bitmap(path + @"
\idiomas\ingles.png");
transparent = new
System.Drawing.Imaging.Image
Attributes();
g.DrawImage(bandera, new
Rectangle(10 * res, posY +
(lenY - 23*res) / 2, 39 *
res, 23 * res), 0, 0,
bandera.Width,
bandera.Height,
GraphicsUnit.Pixel,
transparent);
```

⇔

```
Bitmap bandera;
id =
getResources().getIdentifier("inglesb",
"drawable",  Global.paquete);
banderas[0] =
BitmapFactory.decodeResource(getResource
s(), id);
pintar = new Paint();
g.drawBitmap(Bitmap.createScaledBitmap(b
andera,39,26,true), 10, posY + (lenY -
26) / 2,pintar);
```

Aunque parece hasta ahora la regla más compleja, en realidad es una de las más sencillas y suele estar basada en la última línea de ambos códigos. Las cosas que se deben tener en cuenta son:

- La creación de los objetos imagen (Image o Bitmap) se realiza al arrancar la aplicación, esto implica un gasto de memoria, que podemos considerar despreciable teniendo en cuenta la calidad de las imágenes, su reutilización y las pocas pantallas que tiene la aplicación. En aplicaciones con más carga esta información se debe cargar y liberar antes y después del pintado.

- Las rutas a los archivos se hace de manera diferente para ambas plataformas, en Windows Mobile se accede a la ruta del archivo igual un ordenador, mientras que en Android obtenemos un identificador único para un recurso de la aplicación que podemos utilizar para cargar la imagen.

- Como se puede observar en ambos métodos la imagen se pinta escalada al tamaño que nosotros deseamos, y que para ello se tendrá en cuenta el tamaño y la densidad de pantalla de los sistemas operativos.

Teniendo en cuenta que el proceso completo de carga y pintado de imagen no se encuentra unificado en una sola parte del código, para este caso no se ha realizado ningún método alternativo para su uso.

4.5.1.4 Descarga de archivos

En los diferentes lenguajes de programación existen librerías para la comunicación con la red.

El código se puede encapsular dentro de un método y dotarlo de más funcionalidad, o usarlo directamente. En Windows Mobile:

```
HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);
req.Timeout = 30000;
req.KeepAlive = false;
WebResponse result = req.GetResponse();
Stream rs = result.GetResponseStream();
BinaryReader br = new BinaryReader(rs);
FileStream fs = File.Open(path, FileMode.Create);
BinaryWriter bw = new BinaryWriter(fs);
byte[] buffer = new byte[512];
int readChar;
while ((readChar = br.Read(buffer, 0, buffer.Length)) > 0)
{
   bw.Write(buffer, 0, readChar);
}
rs.Close();
br.Close();
bw.Close();
fs.Close();
```

En Android:

```
URL u = new URL(url);
HttpURLConnection c = (HttpURLConnection) u.openConnection();
c.setRequestMethod("GET");
c.setReadTimeout(30000);
c.setDoOutput(true);
c.connect();
File fichero = new File(path);
InputStream in = c.getInputStream();
OutputStream out = new FileOutpuStream(fichero);
byte[] bytes = new byte[512];
for (int c = in.read(bytes); c != -1; c = in.read(bytes)) {
      out.write(bytes, 0, c);
}
in.close();
out.close();
```

Como se puede observar ambos códigos son muy parecidos, en ambos se establece una conexión tipo "*GET*" para obtener los datos de una URL y guardarlos a nivel de byte. También se puede observa que se ha establecido un *TimeOut* de conexión de 30 segundos para asegurarnos de intentar obtener la información aunque nos encontremos con una conexión lenta tipo GPRS. Como se ha nombrado anteriormente el sistema de ficheros de Windows Mobile es estándar y se puede usar como se desee, en cambio en Android deberemos asegurarnos de meter los ficheros en la ruta `"/data/data/"+ paquete +"/files/"`, donde "paquete" es el nombre de paquete (*PackageName*) de nuestra aplicación Android, y es el lugar al que la aplicación tiene acceso para gestionar ficheros.

4.5.1.5 Eventos táctiles

En los diferentes sistemas operativos tenemos métodos que nos indican cuando se ha producido un evento táctil, estos métodos deberemos sobrescribirlos para acomodarlos a nuestras necesidades, la plantilla base utilizando un sistema de estados para cada pantalla se mostrará a continuación.

En Windows Mobile:

```
private void frmPrincipal_MouseDown(object sender, MouseEventArgs e)
{
   movimiento = false;
   punteroX = e.X;
   punteroY = e.Y;
   switch (estado)
   {
      case Estado.carga:
         break;
      case Estado.principal:
         break;
   }
}

private void frmPrincipal_MouseMove(object sender, MouseEventArgs e)
{
   punteroX = e.X;
   punteroY = e.Y;
   switch (estado)
   {
      case Estado.carga:
         break;
      case Estado.principal:
         break;
   }
}

private void frmPrincipal_MouseUp(object sender, MouseEventArgs e)
{
   punteroX = e.X;
   punteroY = e.Y;
   switch (estado)
   {
      case Estado.carga:
         break;
      case Estado.principal:
         break;
   }
   Thread.Sleep(100);
}
```

En Android:

```
@Override
public boolean onTouchEvent(MotionEvent e) {
       int action = e.getAction();
       int punteroX =(int)e.getX();
       int punteroY =(int)e.getY();
       if (action == MotionEvent.ACTION_DOWN) {
```

```
                movimiento = false;
                switch (estado)
                {
                        case carga:
                          break;
                        case principal:
                          break;
                }
        }

        if (action == MotionEvent.ACTION_UP) {
                switch (estado)
                {
                        case carga:
                          break;
                        case principal:
                          break;
                }
        }

        if (action == MotionEvent.ACTION_MOVE) {
                switch (estado)
                {
                        case carga:
                          break;
                        case principal:
                          break;
                }
                try{
                   Thread.sleep(100);
                }
                catch(Exception e2){}
        }
        // return super.onTouchEvent(event);
        return true;
}
```

En esos códigos base se pueden observar varias cosas, que se van a comentar a continuación:

- En el ejemplo base sólo se están poniendo estado principal y carga, pero se tendrá un estado por cada pantalla que tenga la aplicación.
- La variable movimiento sirve para el uso del desplazamiento de las listas, de forma que podamos detectar si el usuario está tocando la pantalla o ya la ha soltado. Solo se activará en los estados que se requiera.
- La espera que se introduce después de soltar la pantalla impide errores de sincronismo por el posible efecto de que se pulse dos veces sin desearlo la pantalla. Es importante recordar que antes del 2010 la mayoría de las pantallas eran resistivas y los sistemas operativos no se encargaban de filtrar estos efectos no deseados, de hecho era de uso común tener punteros en lugar de usar los dedos para tocar las pantallas.

4.5.2 Regla 1: Cabeceras de librerías

En los diferentes leguajes de programación existen diferentes maneras de indicar a las clases las librerías que se van a utilizar. En nuestro caso utilizamos C# y Java.

Para facilitar el proceso de migración lo mejor es identificar todas las librerías que nos van a hacer falta en los diferentes objetos, de esta manera podemos incluir la misma cabecera en todos las clases de nuestro proyecto.

Un ejemplo sería:

```
using System;                    ⇔      import java.util.Date;
using System.Threading;                 import java.util.Timer;
```

En Android si se desea limpiar la cabecera de librerías, eclipse proporciona un sistema de auto detección de librerías: simplemente pulsando la combinación de teclas “Shift+Ctrl+o”.

4.5.3 Regla 2: Namespace/Package

En los diferentes lenguajes de programación existen diferentes maneras de empaquetar un grupo de objetos, en concreto para Windows Mobile se utiliza Namespace, y para Android se utiliza package.

Esta regla se puede utilizar en combinación con la anterior, puesto que siempre aparece en la parte superior de los archivos junto con la importación de librerías.

Por ejemplo:

```
namespace                        ⇔      package
Pocket_World_Cup_2010                   ricardofraile.android.proyectos.worldcup;
{
…
}
```

4.5.4 Regla 3: Listas y Arrays

La manera de implementar las listas entre C# y Java inicialmente es muy parecida, en realidad cambia su constructor, sus métodos y la manera de acceder a sus elementos.

Una manera de solventar este problema es la utilización de *arrays* en su lugar cuando es posible, que en ambos lenguajes se utilizan exactamente de la misma manera, exceptuando el parámetro “*Length*”, que en Windows Mobile se utiliza con mayúscula y en Android con minúscula. Por ejemplo, si con un *parser* XML leemos un número de objetos indeterminado, justo al final de la lectura podemos crear un *array* de ese tamaño y transferir todos los objetos. A partir de ese punto el código a migrar es mucho menor.

Un ejemplo donde se ven todas las transformaciones que se tienen que hacer con las listas sería el siguiente:

```
List<String> lista = new                ⇔   List<String> lista = new
List<String>();                             ArrayList<String>();
String s = "test";                          String s = "test";
lista.Add(s);                               lista.add(s);
String temp="";                             String temp="";
for(int i=0;i<lista.Count;i++)              for(int i=0;i<lista.size();i++)
   temp = temp+lista[i];                       temp = temp+lista.get(i);
```

4.5.5 Regla 4: Equals/equals

El método de comparación de objetos genérico para ambos lenguajes se utiliza exactamente de la misma manera, pero en C# el método se nombra con mayúscula y en Android con minúscula.

Un ejemplo de la transformación:

```
if(estado.Equals(Estado.idioma))   ⇔   if (estado.equals(Estado.idioma))
```

4.5.6 Regla 5: DateTime/Date

La clase utilizada para las fechas en las diferentes plataformas también debe transformarse, mientras que en Windows Mobile utilizamos la clase DateTime, en java utilizamos la clase Date.

Un ejemplo de transformación:

```
DateTime d = new DateTime(anyo,    ⇔   Date d=new Date(anyo-1900,mes-
mes, dia, hora, minuto,                1,dia,hora,minuto,segundo);
segundo);
```

4.5.7 Regla 6: Strings

La manera de definición y uso de cadenas de textos en ambas plataformas se hace de la misma manera, pero los métodos relacionados con las cadenas de texto no se utilizan o nombran de la misma manera.

Un ejemplo de transformación:

```
String fecha =partidos[0].fecha;   ⇔   String fecha = partidos[0].fecha;
String hora =                          String hora =
partidos[0].hora.Substring(0,2);       partidos[0].hora.substring(0, 2);
String minuto =                        String minuto =
partidos[0].hora.Substring(3,2);       partidos[0].hora.substring(3, 5);
…                                      …
if (temp.IndexOf("(") > 0)             if (temp.indexOf("(") > 0)
```

4.5.8 Regla 7: int/Integer

Cuando transformamos una cadena de texto en un número entero se realiza de manera diferente si usamos C# o Java, en Windows Mobile se transforma directamente utilizando un método de int mientras, mientras que en Java utilizamos la clase Integer.

Un ejemplo de uso:

```
int.Parse(fecha)                    ⇔    Integer.parseInt(fecha)
```

4.5.9 Regla 8: Timers

La utilización de los Timers para utilizar tareas en determinados momentos o de forma repetitiva también varía entre ambos lenguajes.

Un ejemplo de uso:

```
timerRefrescoPartidos = new
System.Threading.Timer(actuali
zarPartidos, this, 0, 5000);
```

⇔

```
timerRefrescarPartidos = new
Timer();
timerRefrescarPartidos.schedule(new
TimerTask() {
   @Override
   public void run() {
      actualizarPartidos();
   }
}, 0, 5000);
```

4.5.10 Regla 9: Streams

La manera de en que se leen y se escriben los ficheros cambia entre ambas plataformas, con un uso parecido de los Streams de datos. Mientras en Windows Mobile se usa StreamReader/StreamWriter, en Android se usa BufferedReader/BufferedWriter.

Un ejemplo de uso:

```
StreamReader sr=new
StreamReader(path);
String line=sr.ReadLine();
while(line!=null) {
   String partido_id=sr.ReadLine();
   …
   partidoss.Add(temp);
   line=sr.ReadLine();
}
sr.Close();
```

⇔

```
BufferedReader sr = new
BufferedReader(new FileReader(path));
String line = sr.readLine();
while (line != null) {
   String partido_id = sr.readLine();
   …
   partidos.add(temp);
   line = sr.readLine();
}
sr.close();
```

4.5.11 Regla 10: XMLs

Hay muchas maneras de leer archivos XMLs, en este caso he elegido usar bases de datos por simplificar el proceso de lectura de los archivos.

Para Windows Mobile usamos DataSet, mientras que en Android usamos Document, en ambos casos el archivo xml se mapea como una base de datos en la memoria.

Un ejemplo de uso:

```
partidosDS = new
DataSet("Partidos");
partidosDS.ReadXml(path +
@"\PartidosWC10.xml");
numeroPartidos =
partidosDS.Tables["Partido"].R
ows.Count;
partidostemp = new
Partido[numeroPartidos];
for (int i = 0; i <
numeroPartidos; i++)
{
   String id =
partidosDS.Tables["Partido"].R
ows[i]["id"].ToString();
   …
}
```

⇔

```
DocumentBuilder builder =
datosDBbuilder.newDocumentBuilder();
partidosDS = builder.parse(new
FileInputStream(fichero));
numeroPartidos =
partidosDS.getElementsByTagName("Par
tido").getLength();
partidostemp = new
Partido[numeroPartidos];
lista =
partidosDS.getElementsByTagName("Par
tido");
for (int i = 0; i < numeroPartidos;
i++)
{
   Element e=(Element)lista.item(i);
   String id =
e.getElementsByTagName("id").item(0)
.getFirstChild().getNodeValue().toSt
ring();
   …
}
```

4.5.12 Regla 11: Mensajes

Las plataformas móviles que usamos tienen un sistema para mostrar mensajes/diálogos de sistema, de manera que podamos mostrar información importante y bloqueante sin necesidad de tener que preocuparnos como programarlo.

Para Windows Mobile utilizamos MessageBox y para Android AlertDialog

Un ejemplo de uso sería:

```
MessageBox.Show(idioma.actua
lizado);
```

⇔

```
AlertDialog dialogo = new
AlertDialog.Builder(act).create();
dialogo.setTitle(idioma.actualizar);
dialogo.setButton("OK", new
DialogInterface.OnClickListener() {
   public void onClick(DialogInterface
dialog, int which) {
   return;
   }
});
dialogo.setMessage(idioma.actualizado);
```

```
dialogo.show();
```

4.5.13 Regla 12: Exit

Las plataformas móviles que utilizamos tienen un método que sirve para cerrar la aplicación y liberar los recursos. Es importante utilizar estos métodos, porque por norma general tanto Windows Mobile como Android, mantienen la aplicación en segundo plano cuando se sale de ella.

Un ejemplo de uso:

```
Application.Exit();                ⇔    System.exit(0);
```

4.5.14 Regla 13: Multimedia

Las plataformas móviles que utilizamos tienen librerías para utilizar los sistemas multimedia del dispositivo. En nuestro caso lo utilizamos para reproducir sonidos cuando se produce un gol.

Para Windows Phone esta librería es System.media.SoundPlayer y para Android la librería Android.media.MadeiaPlayer.

Un ejemplo de uso:

```
reproductor = new                  ⇔    reproductor =
SoundPlayer(path +                      MediaPlayer.create(context,
@"\sonidos\gol.wav");                   R.raw.gol);
reproductor.Load();                     reproductor.start();
reproductor.Play();
```

4.6 Aplicación de las reglas de transformación

Para aplicar las reglas de transformación debemos partir de un proyecto en una de las dos plataformas, cualquiera de las dos analizadas, preferiblemente terminado.

Recordemos que lo que se busca es ahorrar tiempos de desarrollo, cualquier cambio realizado antes de llevar el proyecto a la nueva plataforma se realizará sólo una vez, en el momento que ya se ha transformado el código, disponemos de 2 proyectos independientes, de forma que en lugar de hacer un solo cambio, se tendrán que realizar en ambas plataformas.

El proceso completo sería el siguiente:

1- Desarrollar y terminar proyecto en plataforma origen

2- Crear proyecto nuevo en plataforma destino

3- Copiar recursos gráficos y multimedia

4- Copiar los archivos de código cambiando la extensión del archivo, en Android los archivos son .java y en Windows Mobile son .cs

5- Aplicar las reglas de transformación

6- Añadir nuevas reglas de transformación para uso futuro si fuese necesario.

7- Verificar posibles incompatibilidades

8- Revisar el código para corregir ineficiencias

9- Modificar el código que se desee en los lugares donde el resultado no sea óptimo

El punto 6 es importante, las reglas de transformación que se exponen en este proyecto son las que se han detectado al tratar el código, y según el tipo de proyecto que se realice nos podemos ir encontrando con otros tipos de códigos, como por ejemplo uso de la vibración del dispositivo, añadiendo la regla pertinente.

Otro punto interesante de mencionar es el 7, no todos los dispositivos, ni las plataformas ofrecen las mismas opciones, por lo que nos podemos encontrar con códigos que no puedan llevarse de una plataforma a otra, un ejemplo de esto sería la implementación de funciones asociadas a la tecla física "back" de obligada presencia en cualquier dispositivo Android (con sistema operativo inferior a Android 4.0), y su migración a dispositivos anteriores a Windows Mobile inferiores a la versión 6.5 del sistema operativo, cuando aun no existía esa tecla.

Algunas de estas reglas se pueden automatizar muy fácilmente mediante el uso de scripts, o como la Regla 6 o la Regla 4 (String/string y Equals/equals) usando directamente la opción buscar y remplazar que existe en los entornos de programación. Otras suelen aparecer de manera puntual y localizada, por lo que es más sencillo remplazarlo manualmente que implementar un script que se encargue de hacerlo, como por ejemplo la Regla 8 o la Regla 11 (Timers y Mensajes).

El punto 8 es una tarea que puede ser costosa, y no es necesaria en sí misma, pero si recomendable. Por ejemplo si se pintan 2 líneas en un trozo de código como una celda, se crea un "Pen" o un "Paint" por cada una aplicando la regla de forma directa, pero en ambas líneas este sería igual, por lo que al revisar el código podemos ir quitando duplicados que no van a impedir que funcione la aplicación correctamente, pero si van a disminuir la calidad del código y su eficiencia.

Se debe recordar que esto es una herramienta que debe permitir ahorrar tiempo y costes en el desarrollo de una aplicación móvil multiplataforma, no es una herramienta que nos migre el proyecto completo y funcione. Para su uso se requiere algunas nociones básicas de Android y de Windows Mobile, y sirve para reutilizar una gran parte del código y reducir los tiempos de desarrollo. Según sea el proyecto, esta herramienta podrá ayudar en un 90% del desarrollo o en un 30%, por ejemplo si sólo algunas pantallas cumplen los requisitos del marco genérico.

5. Caso de Uso: Aplicación del Mundial

El objetivo del proyecto es ahorrar costes en el desarrollo de aplicaciones multiplataforma, concretamente para Windows Mobile y Android, por lo que se va a proceder a desarrollar una aplicación usando el marco genérico como base en una de las dos plataformas (Windows Mobile) y se va a intentar proceder a migrarla a la otra (Android).

Debido a que en el año 2010 tenemos un mundial de fútbol, y que se busca una aplicación que pueda ser usada por una cantidad de usuarios elevado, se ha decidido que la aplicación que se va a diseñar basada en el marco genérico sea sobre esta competición.

Lo primero que vamos a definir es las pantallas que va tener la aplicación y que información debería tener cada pantalla. Los estados de la aplicación se podrán relacionar íntimamente con las pantallas:

- Pantalla de carga: Se carga la información de los XMLs de la memoria y se calcula clasificación y otros datos.
- Pantalla de idioma: Para seleccionar el idioma de la aplicación (alta difusión).
- Pantalla principal: Información resumen de próximos partidos
- Pantalla actualizar: Para poder actualizar los datos de la aplicación.
- Pantalla fases de grupo: Con toda la información de las fases de grupo
- Pantalla fases finales: Información de octavos, cuartos, semifinales, 3º y 4º, y final del mundial.
- Pantalla Partido: Con información de un partido determinado.
- Pantalla lista de equipos.
- Pantalla información del equipo.
- Pantalla directo: Para mostrar la información de los partidos que se juegan en el momento.
- Pantalla salir: Para poder cerrar la información.
- Pantalla Acerca de: Información sobre la aplicación.

Deberemos tener métodos para poder pasar entre las diferentes pantallas de manera intuitiva, de forma que el usuario pueda acceder a toda la información.

5.1 Interfaz principal

Como comentamos anteriormente el diseño de la aplicación se debe basar en un concepto ancho fijo y diferentes largos, para poder soportar todas las resoluciones del mercado. Para ello el diseño de la interfaz principal de la aplicación se va a construir de la siguiente manera:

- Barra superior donde pueden aparecer títulos y botones.
- Zona intermedia donde se muestra la información.
- Barra inferior donde se pueden ubicar botones.

En la barra inferior pondremos botones como accesos directos a las pantallas más importantes de la aplicación, ya que algunas pantallas se mostraran sólo tras pulsar en algún sitio de las pantallas principales. Por ejemplo si pulsamos el botón información nos aparece una lista de equipos, en la que podemos pulsar cualquier equipo para acceder a la información del equipo seleccionado.

En la Figura 34 podemos ver un diagrama de este tipo de interfaz.

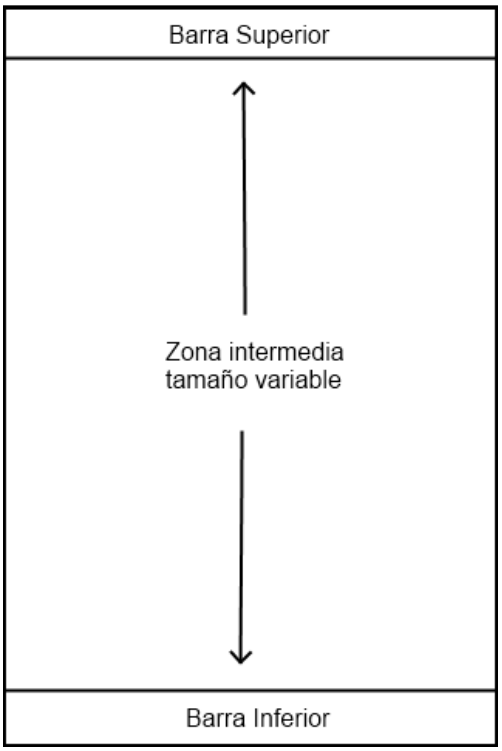

Figura 34: Diseño general de pantalla

5.2 Diseño de las pantallas

A continuación describiremos como debe ser el diseño aproximado de cada una de las pantallas, y que gestos o botones debe reconocer. Para ello primero tomamos la decisión de los botones que van a ir en la botonera inferior, la cual a partir de ahora vamos a denominar como menú de la aplicación:

- Botón acceso a actualizar
- Botón acceso a la fase de grupos
- Botón acceso a la fase de grupos
- Botón acceso a la información de equipos
- Botón acceso al directo
- Botón acceso a salir

Además en la botonera superior, en la parte derecha, tendremos un botón volver en casi todas las pantallas, que al pulsarlo nos llevará a la pantalla de inicio.

5.2.1 Pantalla de carga

En esta pantalla se mostrará el logo de la aplicación, un mensaje animado que indique que está cargando la información, y la información de la versión actual de la aplicación. Esta pantalla no tendrá ningún botón, debido a que no se permite ninguna acción mientras los datos no están cargados.

Mientras que se muestra la pantalla de carga la aplicación se encarga de cargar los siguientes datos:

- Cargar algunas imágenes de uso frecuente en memoria
- Construir los botones de la interfaz principal
- Cargar la información de los equipos
- Cargar la información de los partidos
- Crear las fases y asignar los equipos y partidos pertinentes

Hay que tener en cuenta que en ningún momento se descarga un archivo con la información de las fases o la clasificación, es la propia aplicación la que deberá aplicar las reglas del juego para cada fase, y decidir que equipos pasan a las fases siguientes. Esto aumenta el coste computacional, pero reduce la cantidad de datos a descargar de la red.

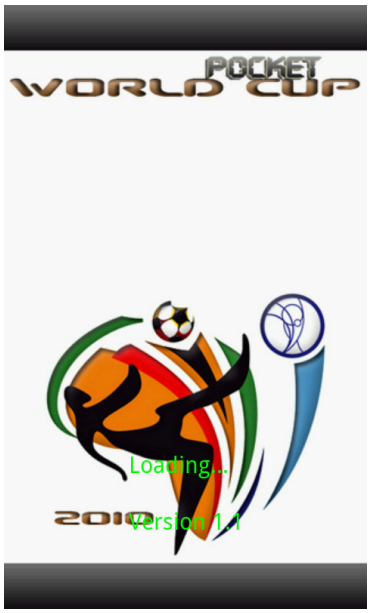

Figura 35: Diseño pantalla de Carga

5.2.2 Pantalla principal

En esta pantalla se mostrará en la parte inferior el menú con los accesos directos a las pantallas principales. Mostrará una cuenta atrás al próximo o próximos partidos que se van a disputar en la competición.

En la barra superior se mostrará un botón configuración para poder ir a la pantalla de elegir el idioma, y el botón acerca de para saber más información de la aplicación. En esta pantalla no se deben tratar ningún gesto especial, solamente deben ser los botones los que reciban los eventos táctiles.

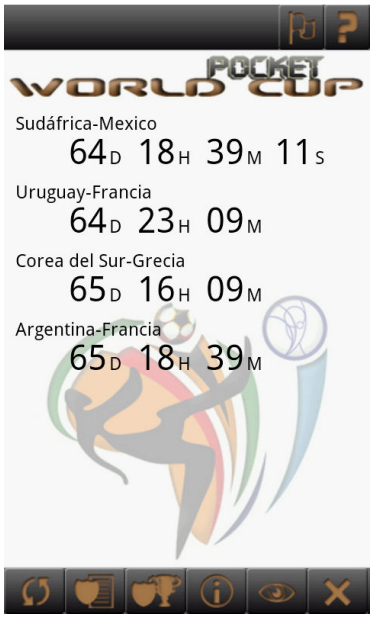

Figura 36: Diseño pantalla principal

5.2.3 Pantalla de idioma

En esta pantalla se mostrará una lista de idiomas en los que se puede utilizar la aplicación. Deberemos proveer a la aplicación de algún sistema sencillo para el cambio de idiomas, de manera que añadir nuevos idiomas tenga un coste reducido.

Está pantalla tendrá el menú en la parte inferior, un botón volver en la barra superior, y un botón guardar debajo de la lista de idiomas, para indicar a la aplicación que hemos cambiado de idioma y recuerde nuestra preferencia para el futuro.

Está pantalla se mostrará cuando se apreté el botón de opciones de la pantalla principal, pero también se mostrará la primera vez que se arranque la aplicación, para obligar al usuario a elegir un idioma y saber cuales están disponibles.

Figura 37: Diseño pantalla selección de idioma

5.2.4 Pantalla actualizar

En esta pantalla se mostrará un botón grande para actualizar los archivos de la aplicación. Deberemos mostrar un mensaje informativo al usuario para que sea consciente de que esta operación requiere conexión y que según su conexión y contrato, esa conexión puede llevar un coste asociado. En la actualidad es muy común que la gente tenga tarifas planas de datos en su Smartphone, pero en el 2010 esta no era una práctica tan común.

Tras pulsar el botón actualizar, si la descarga de los datos se ha hecho correctamente, se pasará a la pantalla de carga para volver a mostrar las información actualizada.

Se proveerá a la aplicación de algún sistema en los archivos, que permita saber si la versión de la aplicación que estamos usando es la más moderna o existe alguna nueva versión de

la aplicación publicada. Esta característica es útil para plataformas en las que las aplicaciones no son instaladas necesariamente desde el *Market*.

En esta pantalla se pintará el menú en la parte inferior y un botón volver en la parte superior, para volver a la pantalla principal.

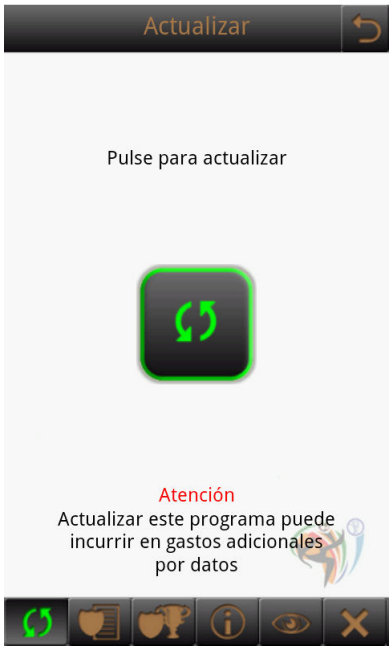

Figura 38: Diseño pantalla actualizar

5.2.5 Pantalla fase de grupos

En esta pantalla se mostrará la información más importante de un grupo de la fase de grupos. En la parte inferior tendremos el menú, y en la superior tendremos el botón volver, dos botones para movernos entre los diferentes grupos, y dos botones para cambiar entre las diferentes vistas:

- Vista clasificación: se ven los equipos del grupos, con los puntos que tiene cada uno y saber cuáles pasarán a la siguiente fase
- Vista partidos: donde se mostrará una lista de todos los partidos correspondientes al grupo.

En esta pantalla se deberán capturar más eventos táctiles:

- *Swipe* a derecha e izquierda para movernos entre los diferentes grupos (sin necesidad de pulsar los botones de la barra superior)
- En la vista clasificación, si se pulsa sobre un equipo, se deberá abrir la pantalla con la información del equipo

- En la vista partidos, si se pulsa sobre un partido, se deberá abrir la pantalla con la información del partido.

- Si alguna de las listas se prevé que pueda ocupar un largo mayor al de la pantalla, deberemos permitir *scroll* de la información.

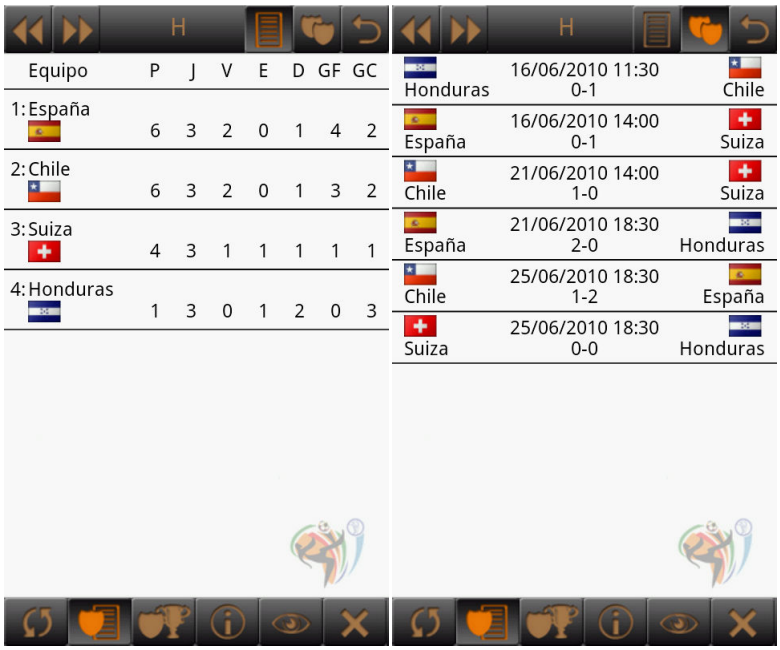

Figura 39: Diseño pantallas fase de grupos

5.2.6 Pantalla fases finales

En esta pantalla se mostrará la información más importante de las fases finales del mundial, es decir, octavos, cuartos, semifinales, lucha por 3º y 4º puesto y la final. En la parte inferior tendremos el menú, y en la superior tendremos el botón volver, dos botones para movernos entre las diferentes fases. En estas fases solo tiene sentido la vista partidos, donde se mostrará una lista de todos los partidos correspondientes a la fase. En esta pantalla se deberán capturar los siguientes eventos táctiles:

- *Swipe* a derecha e izquierda para movernos entre los diferentes grupos (sin necesidad de pulsar los botones de la barra superior)

- Si se pulsa sobre un partido de la fase, se deberá abrir la pantalla con la información del partido.

- Si la lista de la fase se prevé que pueda ocupar un largo mayor al de la pantalla (en octavos al ser ocho partidos, es probable que pase), deberemos permitir *scroll* de la información.

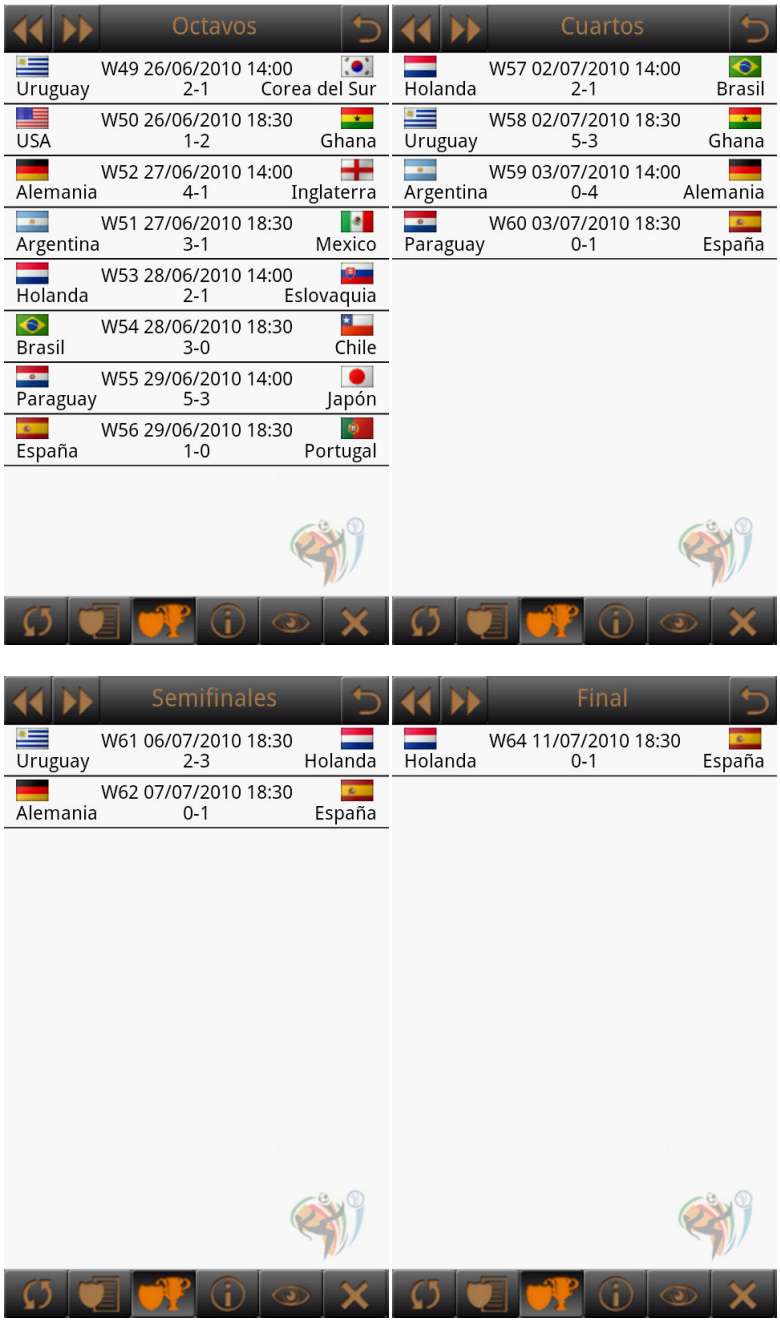

Figura 40: Diseño pantallas fases finales

5.2.7 Pantalla partido

En esta pantalla se mostrará las banderas y nombres de los equipos que disputan el partido y su resultado. Adicionalmente aparecerá una lista de los goles marcados por cada equipo y la persona que ha marcado el gol. En la parte inferior tendremos el menú, y en la superior tendremos el botón volver, dos botones para movernos entre los diferentes partidos:

- Si venimos de la pantalla fases finales, los partidos de la fase correspondiente.
- Si venimos de la pantalla de grupos, los partidos del grupo correspondiente.

En esta pantalla también capturaremos el movimiento de *swipe* para movernos entre los partidos sin necesidad de pulsar en los botones de la barra superior.

Figura 41: Diseño pantalla partido

5.2.8 Pantalla lista de equipos

En esta pantalla se mostrará la lista ordenada alfabéticamente, según el idioma correspondiente, de todos los equipos que participan en la competición junto a su bandera. En la parte inferior se pintará el menú y el botón volver en la parte superior.

Esta pantalla deberá permitir *scroll* para movernos por la lista de los equipos, y al pulsar en alguno determinado, se deberá abrir la pantalla con la información del equipo correspondiente.

Figura 42: Diseño pantalla equipos

5.2.9 Pantalla equipo

En esta pantalla aparecerá la información principal de un equipo. En la parte inferior se mostrará el menú, y en la superior el botón volver y los botones para poder movernos entre los diferentes equipos de la competición.

La información de los equipos que se desea mostrar es:

- Nombre
- Bandera
- Imagen de la plantilla
- Número de mundiales jugados
- Número de mundiales ganados

En esta pantalla se deberá detectar los eventos táctiles tipo *swipe* para poder movernos entres los equipos sin necesidad de usar los botones de la barra superior.

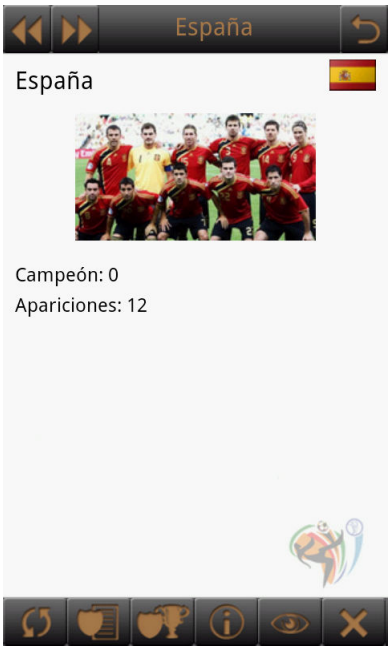

Figura 43: Diseño pantalla información de equipo

5.2.10 Pantalla directo

Esta pantalla se diferencia en dos estados:

- Live apagado (por defecto)
- Live encendido

En ambas pantallas se mostrará el menú en la parte inferior de la pantalla y una serie de botones en la parte superior:

- Volver para volver a la pantalla principal
- Live On-Off para apagar y encender el directo
- Sonido para apagar y encender el aviso sonoro por goles
- Vibración para apagar y encender la vibración por goles

Además, como hemos comentado anteriormente, el factor conectados a la red para bajarnos datos es importante, por lo que cuando el directo esta activo, en el botón directo del menú se verá una animación para que el usuario sea consciente en todo momento de que sigue conectado a la red descargándose datos de manera controlada.

5.2.10.1 Pantalla directo apagado

En esta pantalla beberemos mostrar un mensaje informativo al usuario para que sea consciente de que esta operación requiere conexión y que según su conexión y contrato, esa

conexión puede llevar un coste asociado. Como ya hemos disco anteriormente, en la actualidad es muy común que la gente tenga tarifas planas de datos en su Smartphone, pero en el 2010 esta no era una práctica tan común.

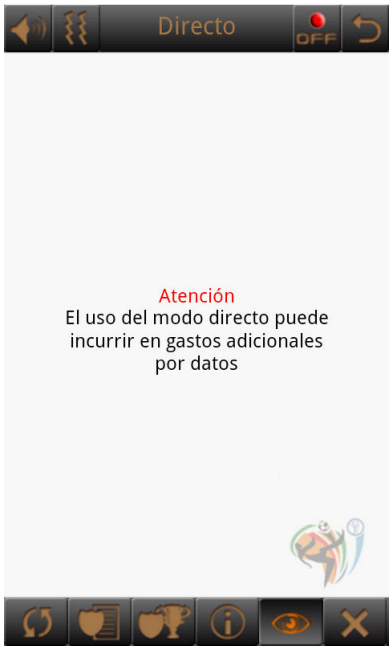

Figura 44: Diseño pantalla directo apagado

5.2.10.2 Pantalla directo encendido

En esta pantalla se verá una lista de eventos que se han producido en los últimos minutos relacionados con los partidos del mundial. Estos eventos tendrán un icono asociado y en algunos casos un aviso sonoro y/o vibración si se encuentran activados.

Los eventos que deseamos detectar, y que dependerán del servidor de datos que utilicemos serán:

- Partido no comenzado: Unos minutos antes de empezar el encuentro, de manera que el usuario sea consciente de que un partido está a punto de empezar
- Primera Parte
- Descanso
- Segunda Parte
- Prorroga (fases finales)
- Penaltis (fases finales)
- Fin del encuentro
- Gol
- Tarjeta amarilla
- Tarjeta roja
- Segunda tarjeta amarilla
- Tarjeta roja a consecuencia de segunda tarjeta amarilla

En las celdas del evento, a parte de un icono representativo del evento, se debería añadir información útil para el usuario, como el resultado, el minuto de partido o el jugador al que se le ha enseñado tarjeta.

Se debe contemplar también que hay eventos que deberán actualizar la información del resto de la aplicación en tiempo real. Por ejemplo tras terminar un partido de una fase se debería actualizar la información de la fase para que la clasificación mostrada este actualizada en todo momento.

Figura 45: Diseño pantalla directo encendido

5.2.11 Pantalla salir

En esta pantalla se mostrará un botón grande para salir de la aplicación. Tras pulsar el botón salir, se liberarán todos los recursos de la aplicación.

En esta pantalla se pintará el menú en la parte inferior y un botón volver en la parte superior, para volver a la pantalla principal.

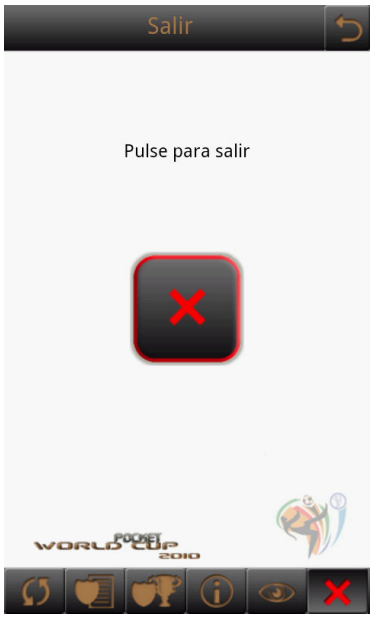

Figura 46: Diseño pantalla salir

5.2.12 Pantalla acerca de

En esta pantalla se pintará información sobre la aplicación, como el autor, el número de versión o el lugar donde se encuentra alojada.

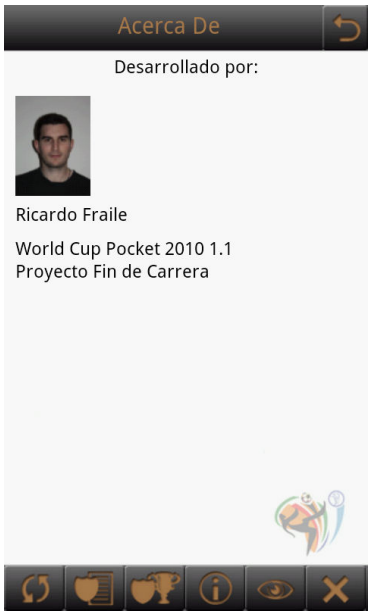

Figura 47: Diseño pantalla acerca de

5.3 Estados de la aplicación

Es fácil suponer que en este tipo de aplicaciones, los estados internos de la aplicación son bastante similares a las pantallas que la aplicación tiene. En la Figura 48 vamos a intentar plasmar el funcionamiento de la aplicación.

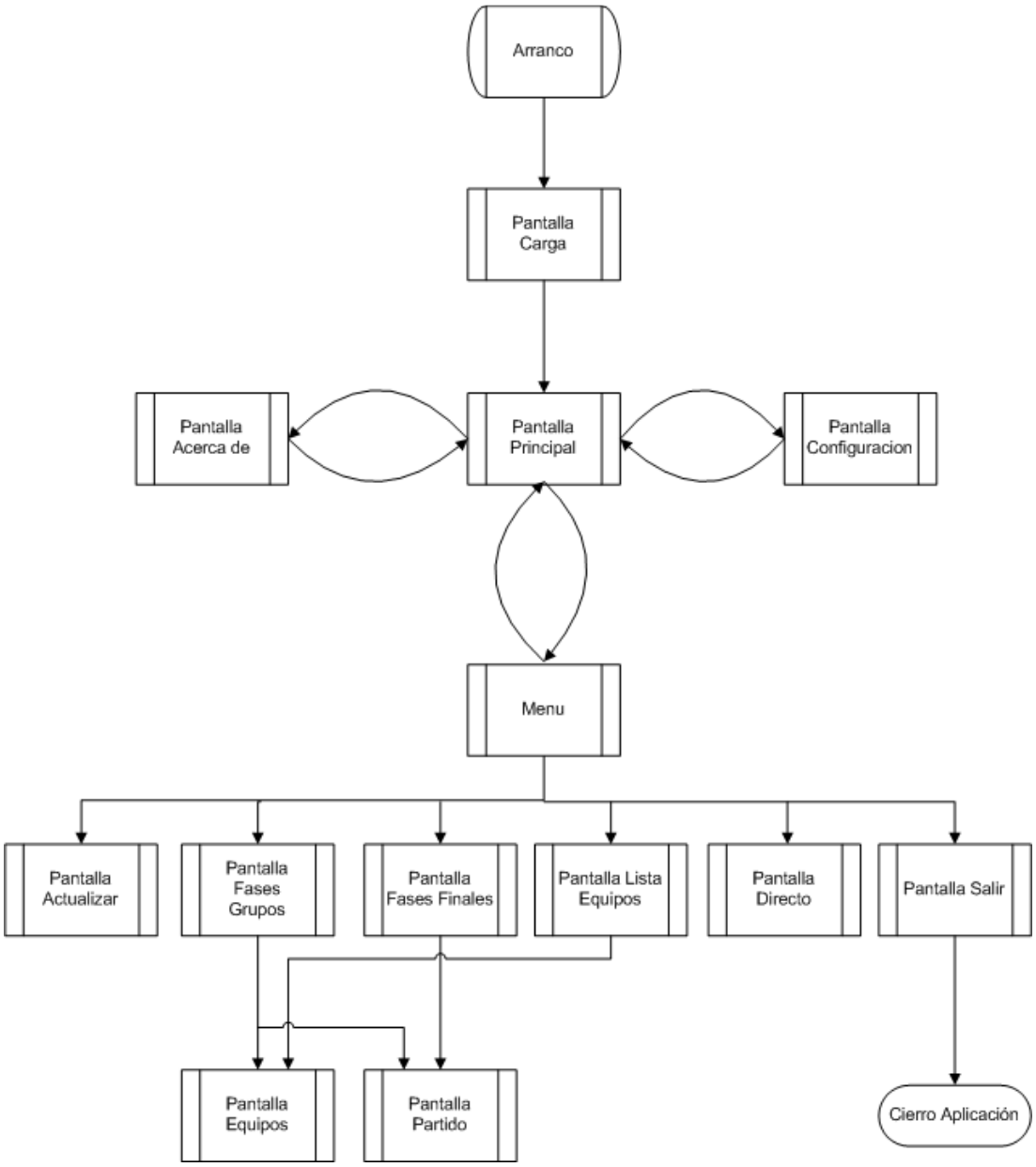

Figura 48: Diagrama de estados de la aplicación

5.4 Interfaz principal

A parte de los criterios de diseño, otro factor a tener en cuenta de las pantallas a la hora de implementar, es la resolución. Debido a que se pretende pintar a bajo nivel, este dato, normalmente viene dado en pixeles, y nos servirá para conocer los límites de la pantalla. Debido a que las pantallas móviles tienen diferentes resoluciones, el marco debe contemplar algún mecanismo de escalado para que las aplicaciones funcionen en el máximo número de dispositivos posibles. El mecanismo utilizado para soportar varias resoluciones de pantalla es el siguiente:

1- Consideramos que la aplicación tiene un ancho fijo.

2- Consideramos que la aplicación puede tener varios largos.

3- Al arrancar la aplicación se detectará el tamaño real de la pantalla y basándonos en el ancho, calcularemos un factor de escalado.

4- Calculamos el largo de la aplicación dividiendo el largo real entre el factor de escalado

5- Diseñamos la aplicación en base a estos valores de ancho y alto.

6- A la hora de pintar en la pantalla a tendremos en cuenta el factor de escalado.

Para entender mejor este proceso, se va a dar un ejemplo típico. Si se supone que se tiene una pantalla HVGA (320x480 pixeles), y se ha decidido decidido usar a nivel de programación un ancho fijo de 240 pixeles (típico de pantallas QVGA, que es una de las resoluciones más pequeñas usadas en las pantallas de dispositivos móviles). Entonces el factor de escalado será:

$$fe=320/240=1,33$$

Y el largo que usaremos a la hora de de definir los límites de la pantalla a la hora de programar será:

$$alto=480/1,33=360$$

Si se desea pintar un cuadrado amarillo en la posición x=10, y=10, con un largo de 20 pixeles, antes de pintarse en la pantalla, el marco modificará la posición y las dimensiones aplicando el factor de escalado, dibujando en la pantalla real un cuadrado amarillo en la posición x=13, y =13 con un largo de 27 pixeles.

Se puede observar que se redondea a un entero el resultado, cometiendo un pequeño error, esto es debido a que para algunos sistemas operativos como Windows Mobile la unidad mínima de pintado es de de un pixel.

Para las fuentes que se utilizan para escribir palabras en la pantalla, se aplicará el mismo el mismo proceso, si se diseña una aplicación que escribe una palabra con un tamaño de letra 8, tras aplicar el factor de escalado, la palabra se pintará con un tamaño de letra 12.

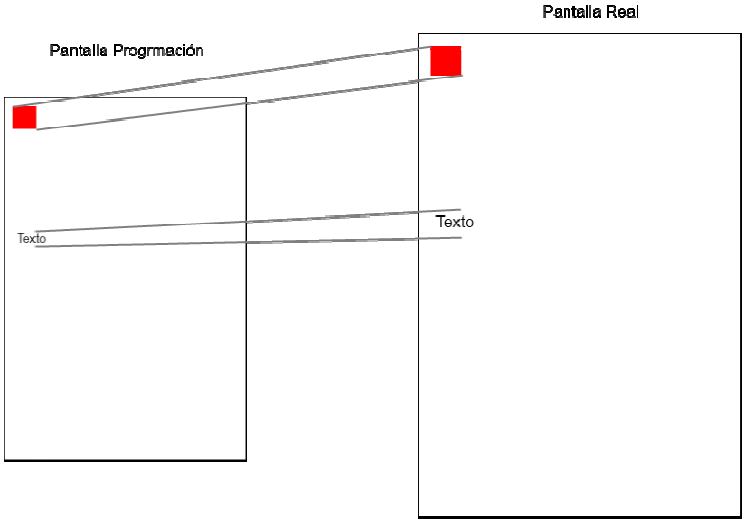

Figura 49: Ejemplo de escalado

Con esto se ha solventado el problema de tener diferentes anchos, pero aun se tienen diferentes largos, este problema se solventa de dos posibles maneras:

- Diseño de una pantalla de información basada en el largo más corto
- Utilización de *scroll* para mover la información de la pantalla de arriba abajo

5.5 Implementación de la aplicación en Windows Mobile

Se va a proceder a crear un proyecto Windows Mobile usando como base el marco genérico que se ha diseñado. Para ello antes de desarrollar cada una de las pantallas se desarrollaran varias componentes de alto nivel que ayudaran al proceso de desarrollo.

5.5.1 Componentes de alto nivel

A continuación vamos a describir las diferentes componentes de alto nivel que se han desarrollado durante la aplicación debido a su uso reiterado, haciendo un código mucho más eficiente.

5.5.1.1 Botón

El botón es un elemento que aparece en cualquier pantalla, y casi se podría decir que la unidad mínima de interacción del usuario con la pantalla. Se ha desarrollado una componente que tiene diferente comportamiento en función de un parámetro tipo:

- Tipo normal: un botón que se pulsa y al soltarse vuelve a su estado normal
- Tipo *check*: un botón que al pulsarse cambia su estado. Muy típico para asociarlo a variables booleanas, donde tenemos un valor true o false.
- Tipo *switch* o selector: cuando tenemos varios botones que están asociados entre sí y solo uno puede estar marcado.

En la aplicación ejemplo que se está desarrollando se usan estos tres tipos de botones en las diferentes pantallas. Para implementar el uso de botones se ha tenido que utilizar:

- Pintar Imagen
- Eventos táctiles

5.5.1.2 Celda

La celda es un elemento que se va a utilizar de forma reiterada en las pantallas que tienen listas, en el caso de esta aplicación, muchas de ellas. Se comportan como un botón de tipo normal, pero en lugar de tener una imagen para pulsado y no pulsado, están compuestas por varios elementos (normalmente textos e imágenes) alineados con algún tipo de orden.

En esta aplicación usamos concretamente cuatro tipos de celdas:

- Celda Idioma: con una imagen y un texto, para mostrar la bandera del idioma y su nombre.
- Celda Clasificación: Con una imagen y ocho textos para mostrar la clasificación de grupos
- Celda partido: Con dos imágenes y cuatro textos
- Celda Evento: Con una imagen y cinco textos para mostrar la información de un evento en directo

Para implementar el uso de celdas se ha tenido que utilizar:

- Pintar Imagen
- Pintar Texto
- Pintar Línea
- Eventos táctiles

5.5.2 Pantalla de carga

Figura 50: Captura Windows Mobile pantalla de carga

Para el desarrollo de la Pantalla de la Figura 50 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto

5.5.3 Pantalla principal

Figura 51: Captura Windows Mobile pantalla principal

Para el desarrollo de la Pantalla de la Figura 51 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior)
- Botón de tipo *switch* (botones del menú inferior)

5.5.4 Pantalla de idioma

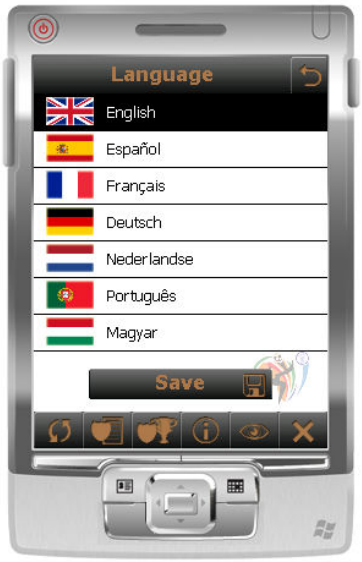

Figura 52: Captura Windows Mobile pantalla selección de idioma

Para el desarrollo de la Pantalla de la Figura 52 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para el uso de la lista

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón guardar
- Botón de tipo *switch* (botones del menú inferior)
- Celdas del tipo idioma

El efecto de movimiento de la lista se ha optado por realizarlo de manera simplificada, es decir, en lugar de desplazarse la lista entera al arrastrar el dedo, el número de celdas permanece fijo en la pantalla y se desplaza el contenido.

5.5.5 Pantalla actualizar

Figura 53: Captura Windows Mobile pantalla actualizar

Para el desarrollo de la Pantalla de la Figura 53 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón central
- Botón de tipo *switch* (botones del menú inferior)

5.5.6 Pantalla fase de grupos

Figura 54: Captura Windows Mobile pantallas fase de grupos

Para el desarrollo de la Pantallas de la Figura 54 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para el uso de las listas de ambas pantallas y para hacer *swipe* entre las diferentes fases de grupos

Y las siguientes componentes de alto nivel:

- Botón de tipo normal.
- Botón de tipo *switch* (botones del menú inferior) y botones de la parte superior derecha que permiten cambiar entre las dos pantallas.
- Celdas del tipo clasificación para una pantalla y del tipo partido para la otra.

5.5.7 Pantalla fases finales

Figura 55: Captura Windows Mobile pantallas fases finales

Para el desarrollo de la Pantalla de la Figura 55 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para el uso de la lista y para hacer *swipe* entre las distintas fases finales del campeonato.

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón guardar
- Botón de tipo *switch* (botones del menú inferior)
- Celdas del tipo partido

5.5.8 Pantalla partido

Figura 56: Captura Windows Mobile pantalla partido

Para el desarrollo de la Pantalla de la Figura 56 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para hacer *swipe* entre los diferentes partidos

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior)
- Botón de tipo *switch* (botones del menú inferior)

5.5.9 Pantalla lista de equipos

Figura 57: Captura Windows Mobile pantalla equipos

Para el desarrollo de la Pantalla de la Figura 57 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para el uso de la lista

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón guardar
- Botón de tipo *switch* (botones del menú inferior)
- Celdas del tipo idioma, pero usando las imágenes de países y sus nombres correspondientes.

5.5.10 Pantalla equipo

Figura 58: Captura Windows Mobile pantalla información de equipo

Para el desarrollo de la Pantalla de la Figura 58 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para hacer *swipe* entre los diferentes equipos

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón guardar
- Botón de tipo *switch* (botones del menú inferior)

5.5.11 Pantalla directo

Figura 59: Captura Windows Mobile pantalla directo encendido y apagado

Para el desarrollo de la Pantallas de la Figura 59 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto
- Eventos táctiles para el uso de la lista de directo encendido

Y las siguientes componentes de alto nivel:

- Botón de tipo normal para el uso del botón volver de la parte superior.
- Botón tipo *check* para los botones de activar directo, el sonido y la vibración.
- Botón de tipo *switch* (botones del menú inferior) y botones de la parte superior derecha que permiten cambiar entre las dos pantallas.
- Celdas del tipo clasificación para una pantalla y del tipo partido para la otra.

5.5.12 Pantalla salir

Figura 60: Captura Windows Mobile pantalla salir

Para el desarrollo de la Pantalla de la Figura 60 se han tenido que utilizar las siguientes primitivas:

- Pintar Imagen
- Pintar Texto

Y las siguientes componentes de alto nivel:

- Botón de tipo normal (botones de la parte superior) y botón central
- Botón de tipo *switch* (botones del menú inferior)

5.6 Implementación de la aplicación Android usando la versión de Windows Mobile

Utilizando como base el proyecto de Windows Mobile se va a proceder a migrar el proyecto a la plataforma Android, para ello vamos a seguir los pasos establecidos y comprobar el resultado.

Se debe mencionar que parte de las reglas mostradas dentro las reglas de transformación se han añadido durante este proceso, al detectar algunas que nos e habían tenido en cuenta. Esto implica que al ir haciendo nuevos proyectos podrán ir apareciendo nuevos reglas que faciliten la migración de nuevas aplicaciones.

Algunas reglas de transformación se aplican al crear los ficheros y no afectan de forma individual al código de cada pantalla. Estas reglas serían:

- De la Regla 0, los eventos táctiles, se aplica al comienzo del proyecto
- Regla 1: Cabecera de librerías
- Regla 2: Namaespace/Package

Cosas que se deben tener en cuenta y que se han mencionado con anterioridad:

- En Android no se usara negrita debido a problemas con las primeras versiones de Android y algunos dispositivos
- Los tamaños y posición de los textos no será exactamente el mismo, sólo aproximado. Las fuentes se usan las estándar del sistema y tampoco son iguales.
- Tras aplicar las reglas y tener el proyecto, se repasa ajustando tamaños de celda y posiciones de forma subjetiva al desarrollador hasta conseguir un resultado deseado. Por ejemplo los tamaños de celda se pueden ajustar teniendo en cuenta el efecto de las fuentes de texto.

5.6.1 Pantalla de carga

Figura 61: Captura Android junto Windows Mobile de pantalla de carga

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 4: equals/Equals

- Regla 6: Strings
- Regla 7: int/Integer
- Regla 8: Timers
- Regla 9: Streams
- Regla 10: XMLs

5.6.2 Pantalla principal

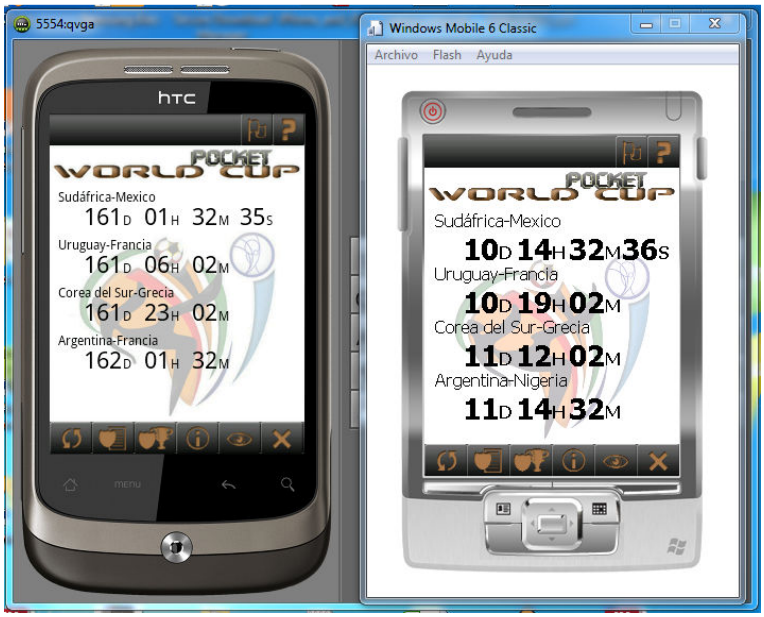

Figura 62: Captura Android junto Windows Mobile de pantalla principal

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 5: DateTime/Date
- Regla 6: Strings
- Regla 11: Mensajes

5.6.3 Pantalla de idioma

Figura 63: Captura Android pantalla junto Windows Mobile de selección de idioma

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 6: Strings

5.6.4 Pantalla actualizar

Figura 64: Captura Android junto Windows Mobile de pantalla actualizar

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 4: equals/Equals
- Regla 6: Strings
- Regla 8: Timers

5.6.5 Pantalla fase de grupos

Figura 65: Captura Android junto Windows Mobile de pantallas fase de grupos

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 5: DateTime/Date
- Regla 6: Strings

5.6.6 Pantalla fases finales

Figura 66: Captura Android junto Windows Mobile de pantalla fases finales

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 5: DateTime/Date
- Regla 6: Strings

5.6.7 Pantalla partido

Figura 67: Captura Android junto Windows Mobile de pantalla partido

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 5: DateTime/Date
- Regla 6: Strings

5.6.8 Pantalla lista de equipos

Figura 68: Captura Android junto Windows Mobile de pantalla equipos

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 6: Strings

5.6.9 Pantalla equipo

Figura 69: Captura Android junto Windows Mobile de pantalla información de equipo

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 6: Strings

5.6.10 Pantalla directo

Figura 70: Captura Android junto Windows Mobile de pantalla directo apagado

Figura 71: Captura Android junto Windows Mobile de pantalla directo encendido

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 3: Listas y Arrays
- Regla 4: equals/Equals
- Regla 5: DateTime/Date
- Regla 6: Strings
- Regla 13: Multimedia

En la Figura 71, se observa que en Windows Mobile se tienen nueve celdas evento mientras que en Android sólo se tienen ocho. Esto es debido como se ha comentado anteriormente a que la resolución de pantalla es QVGA (Android) mientras que la otra es HVGA (Windows Mobile).

5.6.11 Pantalla salir

Figura 72: Captura Android junto Windows Mobile de pantalla salir

Las reglas de transformación que hemos utilizado para migrar el código perteneciente a esta pantalla son las siguientes:

- Regla 0: Reglas del marco
- Regla 4: equals/Equals
- Regla 6: Strings

5.7 Pruebas de rendimiento

Después de tener la aplicación implementada y migrada a ambas plataformas, se debe comprobar que las aplicaciones cumplen con los requisitos de funcionalidad que se deseaban, para ello se van a probar las aplicaciones en los emuladores propios de las plataformas en diferentes dispositivos reales.

5.7.1 Emuladores

Los emuladores de Windows Mobile y de Android son muy prácticos cuando no se dispone de un dispositivo real, o cuando se quieren probar varias resoluciones de una aplicación, pero su comportamiento es muy lento, lo que impide comprobar las usabilidad real de una aplicación.

5.7.2 Dispositivos Reales

Para la realización de las pruebas con dispositivos reales se va a optar por cuatro dispositivos, dos de ellos de características similares para que la comparación de rendimiento entre ambas plataformas se vea afectada lo menos posible por el hardware de los dispositivos.

Lo ideal sería realizar las pruebas con un mismo dispositivo, pero por desgracia los fabricantes diseñan los dispositivos para funcionar con un SO concreto y en las soluciones software, como XAndorid, creadas por la comunidad de desarrolladores no aprovechan toda la potencia del dispositivo, haciendo que su rendimiento sea inferior.

Los dispositivos que se van a utilizar para las pruebas son:

- Windows Mobile: HTC Touch Diamond
- Windows Mobile: HTC HD2
- Android: HTC Wildfire
- Android: Samsung I9000 Galaxy S

Tanto el HTC HD2 como la Sansung Galaxy S disponen de procesadores y memoria muy parecidos, lo que ayudara a valorar el comportamiento, rendimiento y usabilidad de las aplicaciones en los dos sistemas operativos.

Figura 73: Dispositivos de prueba

5.7.2.1 Dispositivo HTC Touch Diamond

Figura 74: Imagen de HTC Touch Diamond

Las especificaciones técnicas del dispositivo ([12]) se pueden observar en la Tabla 2.

General	2G Network	GSM 900 / 1800 / 1900
		GSM 850 / 1800 / 1900 - American versión
	3G Network	HSDPA 900 / 2100
		HSDPA 850 / 1900 - American versión
	SIM	Mini-SIM
	Announced	2008, May. Released 2008, May
	Status	Discontinued
Body	Dimensions	102 x 51 x 11.5 mm (4.02 x 2.01 x 0.45 in)
	Weight	110 g (3.88 oz)
Display	Type	TFT resistive touchscreen, 65K colors
	Size	480 x 640 pixels, 2.8 inches (~286 ppi pixel density)
		- TouchFLO 3D finger swipe navigation
		- Handwriting recognition
Sound	Alert types	Vibration; Downloadable polyphonic, MP3, WAV, WMA ringtones
	Loudspeaker	Yes
	3.5mm jack	No
Memory	Card slot	No
	Internal	4 GB storage, 192 MB RAM, 256 MB ROM
Data	GPRS	Class 10 (4+1/3+2 slots), 32 - 48 kbps
	EDGE	Class 10, 236.8 kbps
	Speed	HSDPA, 7.2 Mbps
	WLAN	Wi-Fi 802.11 b/g
	Bluetooth	Yes, v2.0 with A2DP
	USB	Yes, miniUSB
Camera	Primary	3.15 MP, 2048x1536 pixels, autofocus
	Video	Yes, CIF@30fps
	Secondary	VGA videocall camera
Features	OS	Microsoft Windows Mobile 6.1 Professional
	Chipset	Qualcomm MSM7201A

	CPU	528 MHz ARM 11
	GPU	Adreno 130
	Sensors	Accelerometer, proximity
	Messaging	SMS (threaded view), MMS, Email, Instant Messaging
	Browser	WAP 2.0/xHTML, HTML
	Radio	Stereo FM radio with RDS
	GPS	Yes, with A-GPS support
Battery		Li-Ion 900 mAh battery
	Stand-by	Up to 285 h
	Talk time	Up to 5 h 30 min

Tabla 2. Especificaciones técnicas HTC Touch Diamond

5.7.2.2 Dispositivo HTC HD2

Figura 75: Imagen de HTC HD2

Las especificaciones técnicas del dispositivo ([13]) se pueden observar en la Tabla 3.

General	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 900 / 2100
		HSDPA 850 / 2100 - Australian Version
	SIM	Mini-SIM
	¡Error! Referencia de hipervínculo no válida.	2009, October
	Status	Available. Released 2009, November
Body	Dimensions	120.5 x 67 x 11 mm (4.74 x 2.64 x 0.43 in)
	Weight	157 g (5.54 oz)
Display	Type	TFT capacitive touchscreen, 65K colors
	Size	480 x 800 pixels, 4.3 inches (~217 ppi pixel density)
	Multitouch	Yes
		- Sense UI

Sound	Alert types	Vibration, MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
Memory	Card slot	microSD, up to 32 GB
	Internal	448 MB RAM, 512 MB ROM
Data	GPRS	Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps
	EDGE	Class 12
	Speed	HSDPA, 7.2 Mbps; HSUPA, 2 Mbps
	WLAN	Wi-Fi 802.11 b/g, Wi-Fi router
	Bluetooth	Yes, v2.1 with A2DP
	USB	Yes, microUSB
Camera	Primary	5 MP, 2592 x 1944 pixels, autofocus, dual LED flash, check quality
	Features	Geo-tagging
	Video	Yes, VGA@30fps
	Secondary	No
Features	OS	Microsoft Windows Mobile 6.5 Professional
	Chipset	Qualcomm QSD8250 Snapdragon
	CPU	1 GHz Scorpion
	GPU	Adreno 200
	Sensors	Accelerometer, proximity, compass
	Messaging	SMS (threaded view), MMS, Email, IM
	Browser	WAP 2.0/xHTML, HTML
	Radio	Stereo FM radio with RDS
	GPS	Yes, with A-GPS support; NaviPanel
	Java	Yes, MIDP 2.0
Battery		Li-Ion 1230 mAh battery
	Stand-by	Up to 490 h (2G) / Up to 390 h (3G)
	Talk time	Up to 6 h 20 min (2G) / Up to 5 h 40 min (3G)
	Music play	Up to 12 h

Tabla 3. Especificaciones técnicas HTC HD2

5.7.2.3 Dispositivo HTC Wildfire

Figura 76: Imagen de HTC Wildfire

Las especificaciones técnicas del dispositivo ([14]) se pueden observar en la Tabla 4.

General	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 900 / 2100
	SIM	Mini-SIM
	Announced	2010, May
	Status	Available. Released 2010, May
Body	Dimensions	106.8 x 60.4 x 12 mm (4.20 x 2.38 x 0.47 in)
	Weight	118 g (4.16 oz)
Display	Type	TFT capacitive touchscreen, 16M colors
	Size	240 x 320 pixels, 3.2 inches (~125 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass
		- Optical trackpad
Sound	Alert types	Vibration, MP3
	Loudspeaker	Yes
	3.5mm jack	Yes
Memory	Card slot	microSD, up to 32 GB
	Internal	384 MB RAM; 512 MB ROM
Data	GPRS	Class 10 (4+1/3+2 slots), 32 - 48 kbps
	EDGE	Class 10, 236.8 kbps
	Speed	HSDPA, 7.2 Mbps
	WLAN	Wi-Fi 802.11 b/g, Wi-Fi hotspot (Android 2.2)
	Bluetooth	Yes v2.1 with A2DP
	USB	Yes, microUSB v2.0
Camera	Primary	5 MP, 2592 x 1944 pixels, autofocus, LED flash
	Features	Geo-tagging
	Video	Yes
	Secondary	No

Features	OS	Android OS, v2.1 (Eclair), upgradable to v2.2 (Froyo)
	Chipset	Qualcomm MSM7225
	CPU	528 MHz ARMv6
	Sensors	Accelerometer, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Email, IM
	Browser	HTML
	Radio	Stereo FM radio with RDS
	GPS	Yes, with A-GPS support
Battery		Li-Ion 1300 mAh battery
	Stand-by	Up to 480 h (2G) / Up to 690 h (3G)
	Talk time	Up to 7 h 20 min (2G) / Up to 8 h 10 min (3G)

Tabla 4. Especificaciones técnicas HTC Wildfire

5.7.2.4 Dispositivo Samsung I9000 Galaxy S

Figura 77: Imagen de Samsung I9000 Galaxy S

Las especificaciones técnicas del dispositivo ([15]) se pueden observar en la Tabla 5.

General	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 900 / 1900 / 2100
	SIM	Mini-SIM
	¡Error! Referencia de hipervínculo no válida.	2010, March
	Status	Available. Released 2010, June
Body	Dimensions	122.4 x 64.2 x 9.9 mm (4.82 x 2.53 x 0.39 in)
	Weight	119 g (4.20 oz)
Display	Type	Super AMOLED capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 4.0 inches (~233 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass
		- TouchWiz 3.0 UI

Sound	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
Memory	Card slot	microSD, up to 32 GB
	Internal	8/16 GB storage, 512 MB RAM, 2 GB ROM
Data	GPRS	Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps
	EDGE	Class 12
	Speed	HSDPA, 7.2 Mbps; HSUPA, 5.76 Mbps
	WLAN	Wi-Fi 802.11 b/g/n, DLNA, Wi-Fi hotspot (Android 2.2)
	Bluetooth	Yes, v3.0 with A2DP
	USB	Yes, microUSB v2.0
Camera	Primary	5 MP, 2592 x 1944 pixels, autofocus, check quality
	Features	Geo-tagging, touch focus, face and smile detection
	Video	Yes, 720p@30fps, check quality
	Secondary	Yes, VGA
Features	OS	Android OS, v2.1 (Eclair), upgradable to v2.3 (Gingerbread)
	Chipset	Hummingbird
	CPU	1 GHz Cortex-A8
	GPU	PowerVR SGX540
	Sensors	Accelerometer, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM, RSS
	Browser	WAP 2.0/xHTML, HTML, Adobe Flash
	Radio	FM radio with RDS
	GPS	Yes, with A-GPS support
	Java	Yes, via Java MIDP emulator
Battery		Li-Ion 1500 mAh battery
	Stand-by	Up to 750 h (2G) / Up to 576 h (3G)
	Talk time	Up to 13 h 30 min (2G) / Up to 6 h 30 min (3G)

Tabla 5. Especificaciones técnicas Samsung I9000 Galaxy S

5.7.3 Pruebas

Las pruebas que se van a llevar a cabo para comprobar que la aplicación funciona y que cumple con las expectativas de comportamiento esperadas son las siguientes:

- Funcionamiento en diferentes sistemas operativos
- Funcionamiento en diferentes resoluciones de pantalla
- Tiempos de respuesta

Con cada uno de estos puntos podremos valorar si se ha conseguido que la aplicación utilizada usando el marco genérico y migrada para funcionar en dos sistemas operativos puede ser utilizada por los usuarios de smartphones de forma satisfactoria.

5.7.3.1 Diferentes sistemas operativos

Para probar que la aplicación funciona en diferentes sistemas operativos vamos a proceder a compilarla y generar el ejecutable para cada plataforma.

En Windows Mobile el archivo generado con extensión "`.cab`" se va a proceder a instalar en el dispositivo HTC Touch Diamond, cuya resolución HVGA es igual a la del emulador usado, de forma que su funcionamiento debería ser el mismo.

Tras la instalación la aplicación se ha comprobado que se comporta correctamente, a continuación en la Figura 78 se muestran algunas capturas de la ejecución de la aplicación:

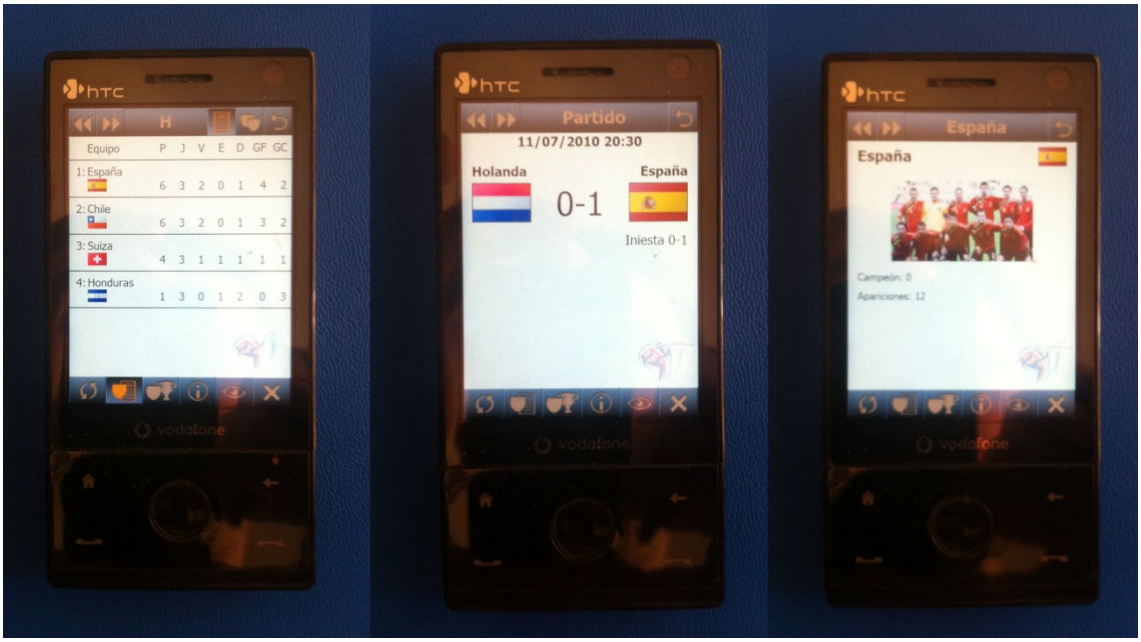

Figura 78: Imagen de la aplicación en un dispositivo Windows Mobile

En Android el archivo generado con extensión "`.apk`" se va a proceder a instalar en el dispositivo HTC Wildfire, cuya resolución QVGA es igual a la del emulador usado, de forma que su funcionamiento debería ser el mismo.

Tras la instalación la aplicación, se ha comprobado que se ha comporta correctamente, a continuación en la Figura 79 se muestran algunas capturas de la ejecución de la aplicación:

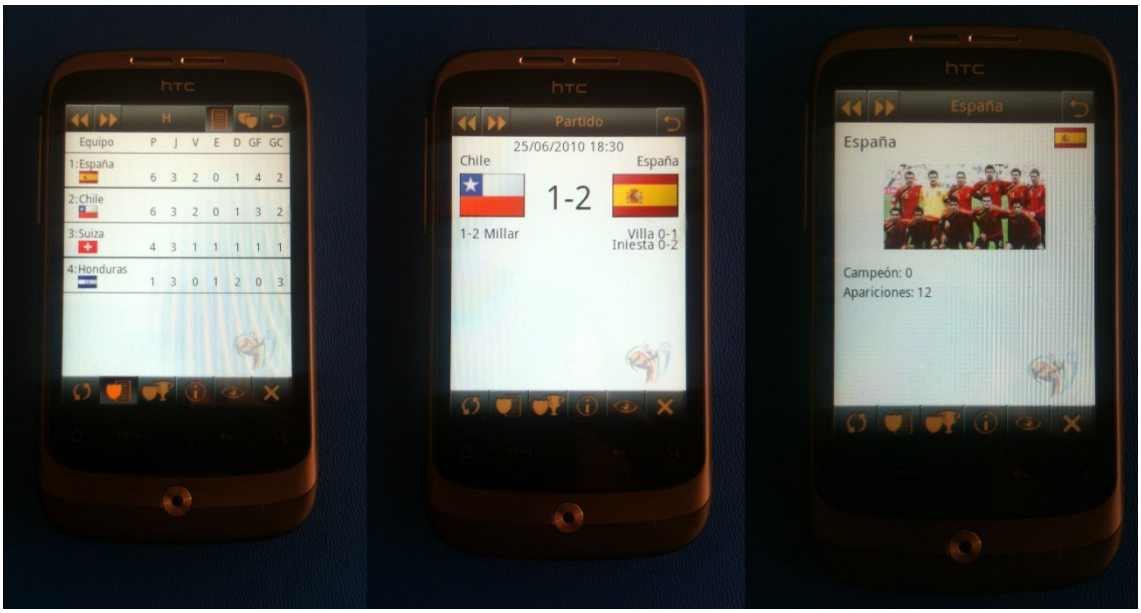

Figura 79: Imagen de la aplicación en un dispositivo Android

5.7.3.2 Diferentes Resoluciones de pantalla

La aplicación se ha creado para funcionar en diferentes tamaños de pantalla, concretamente los más usados en el 2010:

- Windows Mobile: FWVGA, WVGA, VGA, HVGA, QVGA
- Android: FWVGA, WVGA, HVGA, QVGA

Para el funcionamiento correcto de Android, se ha utilizado el parámetro AnyDensity=”false”, de forma que nos aseguramos que la aplicación funciona correctamente a esas resoluciones obligando a ejecutarse con una densidad estándar (por ejemplo en el tamaño de los textos).

Teniendo en cuenta que los dispositivos FWVGA son muy raros (WVGA con 54 pixeles mas de alto), de hecho aparecieron durante un par de años y han vuelto a desaparecer al menos temporalmente, no se dispone de emulador estándar ni de dispositivo real para realizar las pruebas, pero con los comentarios obtenidos de los usuarios tras la publicación gratuita de la aplicación, se sabe que funciona correctamente.

Se procede a instalar la aplicación en la HTC HD2 y se ejecuta en este dispositivo y en la HTC Touch Diamond, y se aprecia que la aplicación se ajusta perfectamente a la pantalla. A continuación en la Figura 80 se muestran un par de capturas.

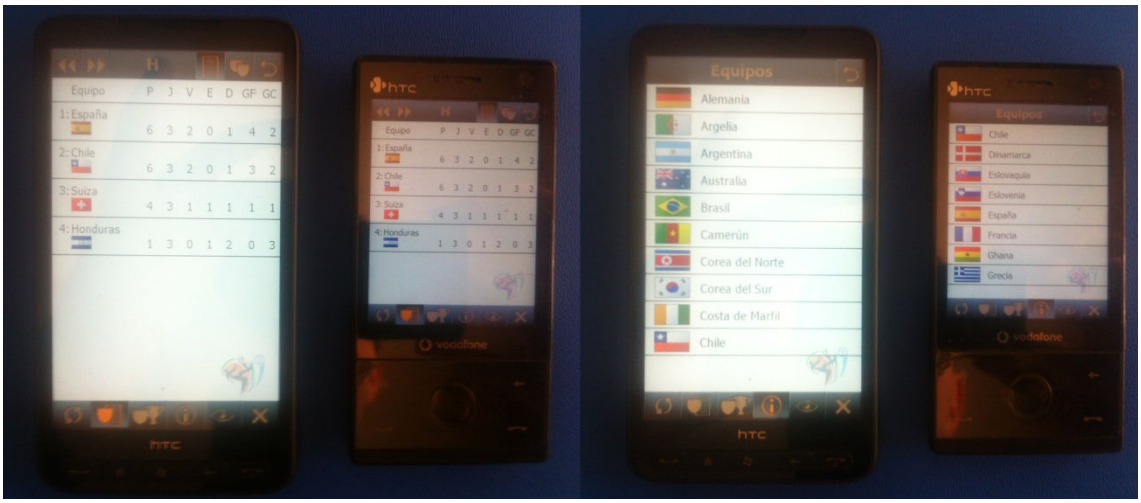

Figura 80: Imagen de la aplicación en ambos dispositivos Windows Mobile

Se procede a instalar la aplicación en la Samsung Galaxy S y se ejecuta en este dispositivo y en la HTC Wildfire, y se aprecia que la aplicación se ajusta perfectamente a la pantalla. A continuación en la Figura 81 se muestran un par de capturas.

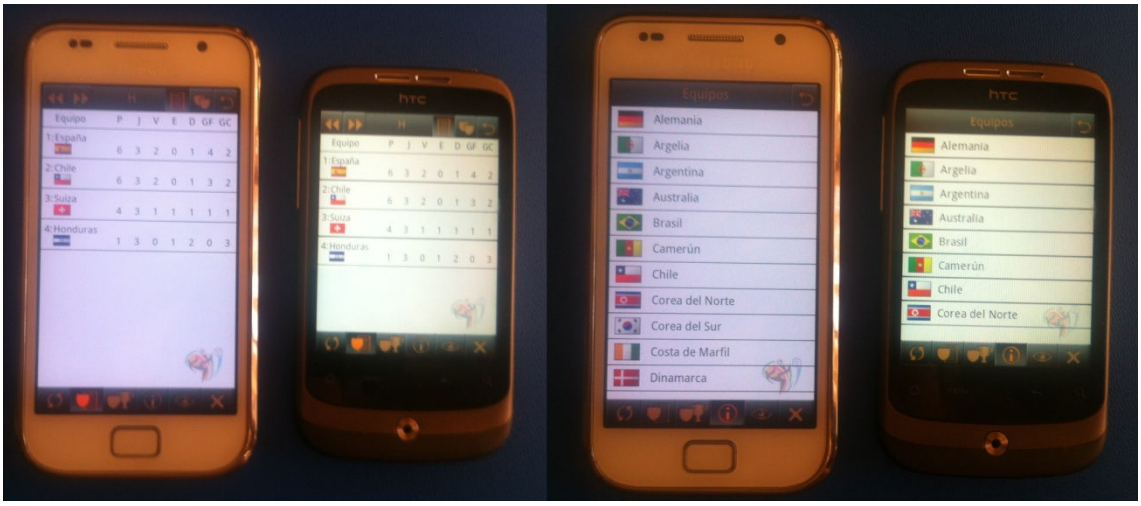

Figura 81: Imagen de la aplicación en ambos dispositivos Android

Por último se procede a comparar la aplicación funcionando en los 4 dispositivos, vemos que el comportamiento es muy similar, siendo mejor en los dispositivos WVGA, que son más modernos y de gama alta. A continuación en la Figura 82 se puede observar una captura.

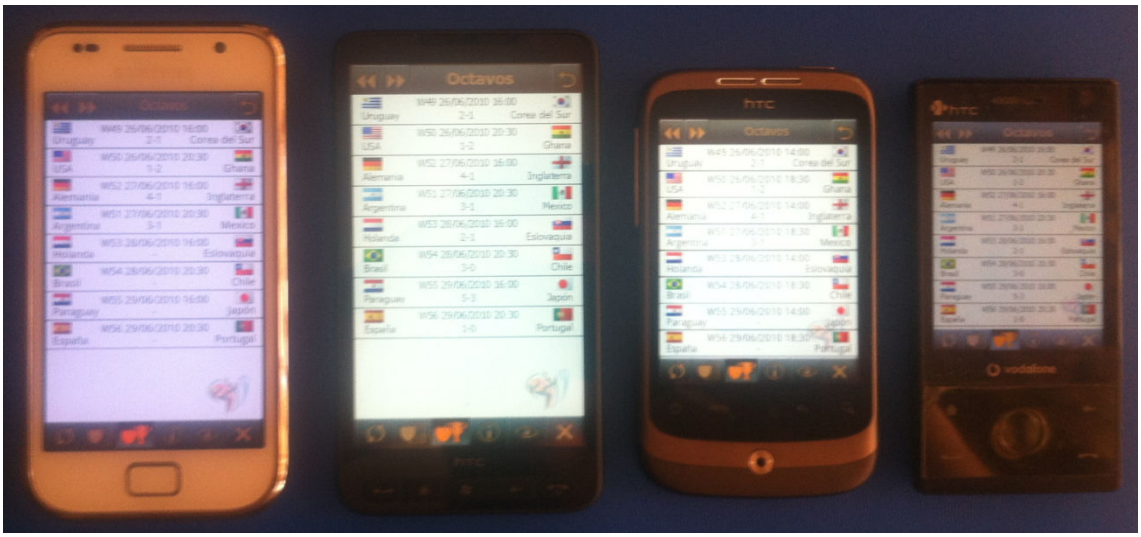

Figura 82: Imagen de la aplicación en todos los dispositivos de prueba

5.7.3.3 Tabla de tiempos

Para tener un tato comparativo de cómo se comporta la aplicación en las diferentes plataformas y dispositivos móviles, vamos a realizar la medición de los tiempos que requieren las tareas más significativas:

- Tiempo de carga: Este el tiempo que tarda la aplicación en cargar todos los datos de los equipos y partidos del mundial de fútbol desde la memoria persistente y calcular la clasificación y demás información que se puede mostrar en la aplicación.

- Tiempo de actualización: Este es el tiempo que tarda la aplicación en bajarse la información de los partidos actualizada desde la web y almacenarla en la memoria persistente para su posterior procesado. La información de los equipos no se vuelve a bajar, puesto que no debe cambiar a lo largo del campeonato.

- Tiempo de refresco del modo live (directo): Este es el tiempo que tarda en descargar el archivo de los eventos que se están produciendo durante un partido y procesarlos. Este archivo es mayor cuantos más eventos hayan pasado durante un partido, siendo un evento los goles las tarjetas, o los tiempos más importantes del partido (inicio, descanso, inicio segunda parte, prorroga, final).

Parte de estos tiempos no sólo dependen del dispositivo, sino del tipo de conexión y del servidor web donde se encuentran almacenados los archivos. Para las pruebas se utilizo una conexión wifi conectada a un router con una conexión de 30 megas y un servidor web gratuito de webcindario, este último es bastante lento a la hora de procesar las peticiones.

El archivo del modo *live* que se ha utilizado para las pruebas es el mismo para todas las pruebas y corresponde a un partido completo, de forma que para ese partido y en estas condiciones el tiempo medido sería mayor que debería alcanzarse a lo largo del partido.

Para medir los tiempos en Windows Mobile se utiliza el siguiente código:

```
DateTime marcaInicial = DateTime.Now;
//Codigo a medir
DateTime marcaFinal = DateTime.Now;
long ms = (marcaFinal.Ticks - marcaInicial.Ticks)
    /TimeSpan.TicksPerMillisecond;
```

En Android el código equivalente para hacer las mediciones sería:

```
Date marcaInicial = new Date();
//Codigo a medir
Date marcaFinal = new Date();
long ms = marcaFinal.getTime() - marcaInicial.getTime();
```

A continuación se muestra la tabla con los tiempos medios obtenidos en los diferentes entornos de pruebas.

	Tiempo de carga (ms)	Tiempo de Actualización (ms)	Tiempo de refresco Live (ms)
Emulador Windows Mobile	14000	6000	3000
HTC Touch Diamond	7000	2000	1000
HTC HD2	1150	880	425
Emulador Android	3000	1000	300
HTC Wildfire	6750	1800	950
Samsung Galaxy S	850	670	350

Tabla 6. Comparativa de tiempos de carga, actualización y refresco del live

Como se puede observar, si se ignoran los tiempos de los emuladores, cuyo comportamiento es atípico, el comportamiento en los cuatro dispositivos es similar, siendo bastante lento en los dispositivos viejos de gama baja HTC Touch Diamond y HTC Wildfire, y siendo bastante bueno en los dispositivos de gama alta HTC HD2 y Samsung Galaxy S.

6. Conclusiones y futuro

En este capítulo final se realiza un análisis del trabajo realizado en este proyecto y se exponen las principales conclusiones alcanzadas tras dicho análisis. A continuación se exponen también los posibles trabajos a realizar de cara a la futura mejora o ampliación del trabajo realizado en este proyecto.

6.1 Conclusión

El objetivo del proyecto era el diseño e implementación de un marco genérico para el desarrollo de aplicaciones móviles, a posteriori aplicárselo al desarrollo de una aplicación en la plataforma Windows Mobile y migrarlo usando unas reglas de transformación a la plataforma Android. Se va a analizar cada uno de los objetivos del proyecto.

1- **Diseño del marco genérico**

Se ha diseñado un marco genérico, que si bien no vale para todas las aplicaciones, es muy útil para muchas de las aplicaciones profesionales del mercado. Con este marco se soportan aplicaciones para presentación de datos en 2D, con capacidad de almacenamiento de información y de uso de la red. Este marco es sólo una base que se puede ir mejorando para soportar más tipos de aplicaciones.

2- **Implementación del marco genérico**

Se ha procedido a identificar unos casos de uso mínimos que debe tener el maro genérico para el diseño de cualquier aplicación, para cada uno de ellos se ha procedido a implementar el código necesario en cada una de las plataformas de desarrollo, concretamente Windows Mobile y Android, y a posteriori se han definido una serie de reglas que permiten migrar el código de una plataforma a la otra.

3- **Implementación de la aplicación**

Se ha desarrollado usando las reglas del marco genérico una aplicación del Mundial 2010 en la plataforma Windows Mobile, para este desarrollo se han implementado componentes de alto nivel como botones o celdas, que han permitido reutilizar mucho código y tener un desarrollo más limpio y eficiente. Posteriormente siguiendo los pasos que se han definido como herramienta para migrar el proyecto, se ha procedido a crea el proyecto en Android

4- Rendimiento

Utilizando los emuladores de las plataformas de desarrollo y dos dispositivos físicos comerciales para cada plataforma se ha comprobado el funcionamiento de la aplicación, consiguiendo el resultado esperado, la aplicación funciona bien en los dispositivos, pero mucho mejor (visualización y tiempos de carga), cuanto mejor es el dispositivo.

Para comprobar si los resultados obtenidos eran realistas, se procedió a publicar la aplicación de manera gratuita al comienzo del mundial de fútbol, teniendo miles de descargas a nivel mundial, y unos comentarios y críticas muy positivos por parte de los usuarios, que incluso se prestaron a mandar traducciones de los textos de la aplicación en hasta una veintena de idiomas, para que su difusión fuese aun mayor.

6.2 Líneas futuras de trabajo

Finalmente se presentan algunas de las posibles líneas futuras de trabajo a seguir de cara a la continuación del trabajo realizado en este proyecto:

- Automatización del proceso creando scripts que se encarguen de hacer las transformaciones de los textos. Igualmente se debe repasar a posteriori el código para la reutilización de variables.
- Creación de listas con desplazamiento real en lugar de desplazamiento del contenido de la celda. Este efecto mejora la visualización de las listas, pero no aporta mejora funcional.
- Mejora del refresco de la aplicación, en lugar de refrescarse de manera constante con un tiempo de refresco, refrescar mediante eventos (tanto del usuario como de la aplicación), optimizando el rendimiento de la aplicación al no tener que estar continuamente repintando.
- Utilización de algunas componentes nativas de los propios sistemas operativos, encapsulándolas dentro de métodos del propio marco genérico. Esta mejora está pensada para sistemas que no sean Windows Mobile, pero sería muy practica en sistemas como iOS, Android y Windows Phone.
- Utilización de pantallas 3D con OpenGL o el motor que corresponda a las diferentes plataformas.

APÉNDICE A. Presupuesto

1.- Autor: Ricardo Fraile Martínez

2.- Departamento: Ing. Telemática

3.- Descripción del Proyecto:

- Título: Marco Genérico para el Desarrollo de Aplicaciones Migrables entre Windows Mobile y Android.
- Duración: 6 meses
- Tasa de costes indirecto: 20%

4.- Presupuesto total del Proyecto (valores en Euros):

13.026,59 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F.	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Basanta Val, Pablo		Analista Programador	0,5	1.539,69	769,85	
Fraile Martínez, Ricardo		Analista Programador	6	1.539,69	9.238,14	
		Hombres mes	**6,5**	**Total**	10.007,99	

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas) Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPO

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Portatil con Win. 7 y Office	1300,00	100	6		130,00
HTC Touch Diamond	400,00	25	6		10,00
HTC HD2	650,00	50	6		32,50
HTC Wildfire	200,00	100	6		20,00
HTC Galaxy S	550,00	100	6		55,00
				Total	247,5

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)

C = coste del equipo (sin IVA)

D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Licencia Visual Studio 2008 Prof.		600,00
	Total	600,00

6

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	10.007,99
Amortización	247,50
Subcontratación de tareas	0,00
Costes de funcionamiento	600,00
Costes Indirectos	2.171,10
Total	**13.026,59**

APÉNDICE B. Glosario

API: (Application Programming Interface) En castellano, interfaz de programación de aplicaciones.

Array: En programación, es una zona de almacenamiento continuo de elementos del mismo tipo.

Check: En programación, un check representa un variable que puede tomar 2 valores diferentes, por ejemplo true-fale o seleccionado-no seleccionado.

CVS: (Concurrent Version System) En castellano, sistema de versionado concurrente, sirve para tener un control de versiones de un código y ayuda a trabajar varias personas sobre un mismo proyecto.

ECJ: (Ecplise Compiler for Java) En castellano, compilador del lenguaje Java para la aplicación Eclipse.

FWVGA: (Full Wide VGA) En castellano, VGA alargado completo o máximo, sirve para denominar la resolución de 854x480 pixeles.

HVGA: (Half-size VGA) En castellano, la mitad de VGA, sirve para denominar la resolución de 320x480 pixeles.

HTTP: (Hypertext Transfer Protocol) En castellano, protocolo de transferencia de hipertexto.

IDE: (Integrated Development Enviroment) En castellano, entorno de desarrollo integrado.

JDT: (Java Development Tools) En castellano, herramientas de desarrollo del lenguaje Java.

Market: Aunque su traducción literal es mercado, referido a smartphones, se refiere al mercado digital donde los desarrolladores publican las aplicaciones y los clientes las pueden descargar (pagando o de forma gratuita).

NET: Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Live: En programación, en vivo o en directo, hace referencia a que refleja algo que está sucediendo en tiempo real (o aproximadamente en tiempo real). Muy usado para eventos deportivos.

OpenGL: (Open Graphics Library) En castellano, librería de gráficos abierta, es una especificación estándar que define un API multiplataforma para escribir aplicaciones con gráficos 2D y 3D.

Parser: Analizador sintáctico, encargado de convertir el texto de entrada en otras estructuras como por ejemplo objetos de un programa.

PDA: (Personal Digital Assistant) En castellano, asistente personal digital. Es una computadora de mano que originalmente pensada como agenda electrónica.

PHP: Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

QVGA: (Quarter VGA) En castellano, un cuarto de VGA, sirve para denominar la resolución de 320x240 pixeles.

RTM: (Release To Manufacturing) En castellano, versión de software que se entrega a los fabricantes, normalmente para que hagan pruebas antes de generar la versión para clientes finales.

Scroll: (desplazamiento) Aplicado a smartphones, acción de mover el dedo en la pantalla en vertical u horizontal para mostrar el contenido oculto.

SDK: (Software Development Kit) En castellano, kit para el desarrollo de software.

Smartphone: Teléfono inteligente, normalmente se utiliza para representar a PDAs con capacidades telefónicas.

Swipe: Aplicado a smartphones, acción de desplazar el dedo de manera rápida horizontalmente en la pantalla, normalmente sirve para pasar de una pantalla a otra como si fueran páginas de un libro.

Switch: En programación, representa una variable que puede tomar varios valores, por ejemplo 1-2-3-4 o azul-verde-rojo.

URL: (Uniform Resource Locator) En castellano, localizador de recursos uniforme.

VGA: (Video Graphics Array) Resolución de 640x480 pixeles, que viene de la tarjeta gráfica comercializada por IBM por primera vez en 1988.

WVGA: (Wide VGA) En castellano, VGA alargado, sirve para denominar la resolución de 800x480 pixeles.

XNA: (Xbox New Architecture) En castellano, arquitectura nueva de Xbox, es un conjunto de herramientas para el desarrollo de juegos para Microsoft.

APÉNDICE C. Referencias e hiperenlaces

[1] PC Actual; "Las ventas de Smartphones siguen rompiendo records", Accesible en Junio de 2013 desde http://www.pcactual.com/2010/11/11/1261/las_ventas_smartphones_siguen_rompiendo_records.html

[2] FIFA; "Copa del Mundo de Fútbol 2010", Accesible en Junio de 2013 desde http://es.fifa.com/worldcup/archive/southafrica2010/index.html

[3] Wikipedia "Windows Mobile", Accesible en Junio de 2013 dese http://es.wikipedia.org/wiki/Windows_Mobile

[4] Wikipedia "C#", Accesible en Junio de 2013 desde http://es.wikipedia.org/wiki/C_Sharp

[5] Wikipedia "Net Compact Framework", Accesible en Junio de 2013 desde http://es.wikipedia.org/wiki/.NET_Compact_Framework

[6] "Android", Accesible en Junio de 2013 desde http://developer.android.com/about/index.html

[7] "Java", Accesible en Junio de 213 desde http://www.java.com/es/download/whatis_java.jsp

[8] Wikipedia "Java", Accesible en Junio de 2013 desde http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[9] "Android SDK", Accesible en Junio de 2013 desde http://developer.android.com/sdk/index.html

[10] "Eclipse", Accesible en Junio de 2013 desde http://www.eclipse.org/

[11] Android "ADT Plugin", Accesible en Junio de 2013 desde http://developer.android.com/tools/sdk/eclipse-adt.html

[12] GSMArena "HTC Touch Diamond", Accesible en Junio de 2013 desde http://www.gsmarena.com/htc_touch_diamond-2368.php

[13] GSMArena "HTC HD2", Accesible en Junio de 2013 desde http://www.gsmarena.com/htc_hd2-2957.php

[14] GSMArena "HTC Wildfire", Accesible en Junio de 2013 desde http://www.gsmarena.com/htc_wildfire-3337.php

[15] GSMArena "Samsung Galaxy S", Accesible en Junio de 2013 desde http://www.gsmarena.com/samsung_i9000_galaxy_s-3115.php